

May 2015

Using Technology to Enhance Opportunities in a Disadvantaged Community through Online Advocacy and Computer Education

Alec Jeffrey Thompson
Worcester Polytechnic Institute

Christopher Henry Murray
Worcester Polytechnic Institute

Kurt Leslie Naugler
Worcester Polytechnic Institute

Follow this and additional works at: <https://digitalcommons.wpi.edu/iqp-all>

Repository Citation

Thompson, A. J., Murray, C. H., & Naugler, K. L. (2015). *Using Technology to Enhance Opportunities in a Disadvantaged Community through Online Advocacy and Computer Education*. Retrieved from <https://digitalcommons.wpi.edu/iqp-all/208>

This Unrestricted is brought to you for free and open access by the Interactive Qualifying Projects at Digital WPI. It has been accepted for inclusion in Interactive Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.



Using Technology to Enhance Opportunities in a Disadvantaged Community through Online Advocacy and Computer Education

Project Centre: Melbourne, Australia D-Term 2015

Submitted by:

Alec Thompson, Chris Murray, Kurt Naugler

banksia-d15@wpi.edu

Submitted to:

Mr Jaime de Loma-Osorio Ricon,

Ms Rachel Wood,

Banksia Gardens Community Services

Project Advisors:

Professor Reinhold Ludwig, Professor Jeanine Skorinko

REL 1502

Date:

May 4, 2015

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see <http://www.wpi.edu/academics/ugradstudies/project-learning.html>

Abstract

Banksia Gardens Community Centre is located in the disadvantaged community of Broadmeadows, Victoria. Broadmeadows residents are often faced with challenges in social areas, as well as in the area of advanced computer skills and technological concepts. This project used information technology to address both of these issues through the development of a series of interactive advocacy videos to raise awareness of violence against women and the implementation of a computer science curriculum for the Community Centre.

Acknowledgments

Our experience in Melbourne, Australia was a once in a lifetime opportunity for us and we would like to extend our gratitude to the following organisations and individuals for making this experience possible.

We would like to thank our sponsor organisation, Banksia Gardens Community Services for providing us with the chance to come to Australia and work in a friendly environment with such good people. We would like to thank Jaime de Loma-Osorio Ricón, Rachel Wood and Jonathan Chee for their advice and support throughout the term. Their commitment to Banksia and the community of Broadmeadows is clear and admirable. We are also grateful to the entire staff of Banksia for welcoming us into their community. The work they do was an inspiration to our whole team.

We would also like to express our gratitude to the members of VCOSS that we were able to meet with and discuss our project. It was very helpful in the completion and understanding of the importance of our project.

For making this opportunity possible we would also like to thank the Worcester Polytechnic Institute Interdisciplinary and Global Studies Division and Professor Holly Ault.

Lastly we would like to thank our advisors from Worcester Polytechnic Institute, Professor Reinhold Ludwig and Associate Professor Jeanine Skorinko for their constant guidance and critical feedback. Their support through the project process was greatly appreciated and imperative for the successful completion of our report.

Executive Summary

Background

Numerous institutions and community organisations use modern technologies, such as websites or computers to aid them in pursuing their mission. Specifically, many of these community organisations have utilized technology in order to address social issues affecting the communities in which they are located. In Victoria, Australia, two of the more prevalent issues include family violence and education. The importance of these two areas is made evident by the fact that they have been listed among the state's top 12 priorities for government expenditure in 2015 (King, 2015). In 2012 there were over 60,000 reported incidents of family violence in Victoria. Moreover, the year 12 graduation rate in Melbourne, the capitol of Victoria, is a mere 55% compared to an 88.1% completion rate across the state (Education and Work, Australia, 2014). In our research, we concentrated on two important issues facing the state of Victoria, resulting in two studies of assessing and enhancing the use of technology by a community organisation.

Banksia Gardens Community Services (BGCS) is a community based, philanthropic organisation which is located in the heart of Broadmeadows. The organisation sponsors many programmes dealing with a range of problems the socially and economically disadvantaged community faces. Two of these programmes, the Good People Act Now (GPAN) youth group and Aiming High after school study programme offer opportunities to integrate technology into these programmes. GPAN deals with the issues of violence against women and gender equality, while Aiming High deals with education in the community.

In the first study we worked with GPAN to integrate technology into its operation in the form of a tailored website with specialised content. The second study involves integrating technology into the Aiming High programme in the form of a computer based, self-taught computer science curriculum.

Study 1: Methodology

In order to study how technology can be used by GPAN, we conducted four focus group meetings with 8 to 15 of the youth group's members to determine the setup and content of the website. To create the website we used the framework provided by WordPress and hosted the site on a laptop during the development stages. Within this site we modified a platform called the Bystander Action Tool and added an educational component in the form of four interactive videos. We developed the interactive videos in collaboration with GPAN and a commercial production company. After completion of the website and the Bystander

Action Tool we were able to survey local community members on the usability and value of content of the site.

Deliverables and Outcomes

We examined the implementation of technology into GPAN by creating a webpage based on user surveys of the group. The webpage contains six pages, including one page hosting the Bystander Action Tool. The surveys showed that local community members thought the webpage was easy to use and informative. Overall, the website was rated an average 5.8 on an ease-of-use scale of seven with seven being easiest to use. When asked about the probability whether viewers would act differently after going through the videos in the Bystander Action Tool, the average response was 5.7; it was rated on a scale of seven, with seven representing very likely to act differently.

Discussion

Through focus groups and working alongside GPAN we identified an opportunity to use technology in a programme operated by a youth organisation. To complete this study we created a webpage for the youth group with the goal to aid them in promoting advocacy for the issue of violence against women. The website and Bystander Action Tool are two pieces of technology that have yielded successful feedback in our surveys.

Study 2: Methodology

To study how technology can be used in the Aiming High programme we developed a computer based curriculum using the programming language Scratch. We conducted semi-structured interviews with three members of BGCS staff who are involved in the Aiming High programme; we asked them questions involving their own computer science knowledge, in addition to the knowledge of Aiming High participants. To create the curriculum we used the programming language Scratch which is a free, open source, visual block based programming language. The curriculum was analysed through surveys of 15 Aiming High students before and after completion of the curriculum. The surveys asked students about the level of concepts presents in the Modules as well as the level of enjoyment of the lesson.

Deliverables and Outcomes

The observations of the group combined with suggestions by the staff members lead our group to testing and implementing the curriculum into the Aiming High programme. The curriculum consists of three modules; a) Introduction to Scratch, b) Module 1: Pong, and c) Module 2: Space Invaders. The idea is that these modules are virtually self-taught so that the student can learn the content of the modules without help of an instructor. In our survey of 15 students (7 male, 8 female) none had any previous programming experience and only one had

a prior interest in computer science. The students who completed the module showed an average increase in familiarity with computer science topics of between 1.125 to 2.375 on a scale of one to seven, with seven representing complete familiarity. The students also showed a slightly increased interest in programming after completing the module. This shows the potential of using technology in an effort to help Aiming High educate its students about programming.

Discussion

In order to attempt to improve the operations of the centre this study focused on one particular educational programme Banksia provides to the community of Broadmeadows. The computer science curriculum currently has the capacity to educate students in the Aiming High programme, as well as participants of any of Banksia's computer programmes, which could equate to over 100 students throughout the year. Surveys of students before and after completion of the modules showed that the modules can be used as an effective tool in teaching young students basic programming principles; concepts they may otherwise never have learned in a community centre.

General Discussion and Conclusion

Through these two studies we explored the use of technology as a tool by a community organisation; we were able to investigate innovative ways that social issues could be targeted with different technologies. BGCS was an ideal study location due to its setting in the heart of a disadvantaged community. The organisation had many programmes that used low-level technology prior to our research. However, for GPAN and the Aiming High programme were determined they would benefit most from the development of technology such as sophisticated website development with interactive video content and an online programming curriculum.

As the world of science advances, there are ever increasing opportunities to utilize new communication technology tools to aid in the operation of community organisations. They result in unique benefits for community organisations such as BGCS. It is for these reasons that we recommend continued research into novel ways that technology can be deployed in different programmes at Banksia. For example, programmes such as the after school study group could use technology in the form of tablets or similar devices to decrease paper consumption by students who complete multiple worksheets during every study group session. This is one of many examples where further research into the use of technology at the centre could bring about positive change in its operation. Our two studies demonstrated an underlying theme that technology can provide a valuable asset for Banksia Gardens

Community Services. It is hoped that our efforts become an enabler to reduce family violence and help reduce unemployment in communities not only in Broadmeadows but other challenged communities worldwide

Table of Contents

ABSTRACT	I
ACKNOWLEDGMENTS	II
EXECUTIVE SUMMARY	III
TABLE OF FIGURES	VIII
LIST OF TABLES	IX
CHAPTER 1: GENERAL BACKGROUND	1
1.1: USING TECHNOLOGY TO ADDRESS SOCIAL ISSUES.....	2
1.1.1: <i>Using Technology to Promote Advocacy</i>	2
1.1.2: <i>Using Technology to Improve Computer Education</i>	3
1.2: FAMILY VIOLENCE IN AUSTRALIA.....	5
1.3: COMPUTER EDUCATION IN AUSTRALIA.....	7
1.4: AN IDEAL STUDY LOCATION: BANKSIA GARDENS COMMUNITY SERVICES.....	9
CHAPTER 2: STUDY 1	11
METHODOLOGY.....	11
DELIVERABLES AND OUTCOMES.....	14
DISCUSSION.....	23
CHAPTER 3: STUDY 2	25
METHODOLOGY.....	25
DELIVERABLES AND OUTCOMES.....	27
DISCUSSION.....	32
CHAPTER 4: GENERAL DISCUSSION	34
CHAPTER 5: CONCLUSION AND RECOMMENDATIONS	35
REFERENCES	36
APPENDICES	40
APPENDIX A.....	40
<i>Website Template</i>	40
APPENDIX B.....	41
<i>Questions for Focus Group with GPAN Members</i>	41
APPENDIX C.....	42
<i>GPAN Website Usability Study: Pre-Survey Commentary and Survey Questions</i>	42
APPENDIX D.....	43
<i>Pre and Post Video Consent Forms</i>	43
APPENDIX E.....	45
<i>Computer Science Curriculum Pre/Post-Module Assessment</i>	45
APPENDIX F.....	46
<i>Getting Started with Scratch</i>	46
APPENDIX G.....	47
<i>Computer Science Module 1</i>	47
APPENDIX H.....	61
<i>Computer Science Module 2</i>	61

Table of Figures

FIGURE 1: RATES OF VIOLENCE AGAINST WOMEN IN AUSTRALIA (DOMESTIC VIOLENCE STATISTICS, 2015) ...	6
FIGURE 2: YEAR 12 GRADUATION RATES IN AUSTRALIA	8
FIGURE 3: HOMEPAGE OF THE NEWLY CREATED GPAN WEBSITE	16
FIGURE 4: GET SMART PAGE	17
FIGURE 5: GET HELP PAGE.....	18
FIGURE 6: GET ACTING PAGE	19
FIGURE 7: GET ALONG PAGE	20
FIGURE 8: BYSTANDER ACTION TOOL PAGE	21
FIGURE 9: PROGRAMME CREATED BY FOLLOWING “GETTING STARTED WITH SCRATCH” GUIDE	29
FIGURE 10: PONG GAME CREATED BY FOLLOWING MODULE 1 INSTRUCTIONS	30
FIGURE 11: CHANGE IN FAMILIARITY OF COMPUTER SCIENCE TOPICS AND PROGRAMMING INTEREST	31

List of Tables

TABLE 1: BYSTANDER ACTION TOOL SCENARIOS AND THEMES	22
TABLE 2: TOPICS AND CONCEPTS COVERED BY EACH MODULE	28

Chapter 1: General Background

In Australia, and more specifically the state of Victoria, located in the southeast region of the country-continent, two prevalent issues that many people are challenged with include family violence and lack of education (A to Z the World, 2015). Family violence is defined to be an act of violent, threatening, or coercive behaviour that occurs within a family. In Victoria there were over 60,000 reported incidents of family violence in 2012 alone (Key Statistics, 2014). In addition, the lack of education is an issue due to low percentages of students completing year 12 and even fewer continuing on to University. This has resulted in a trend of undereducated individuals in the population (Data By Region, 2014). The year 12 graduation rate in Melbourne, the capitol of Victoria, is a mere 55% compared to an 88.1% completion rate across the state. These two issues are so prevalent in Victoria that the state government has made both, family violence and education for children, two of its top 12 priorities for spending in 2015 (King, 2015).

Due to its rapid expansion in the past few decades, technology, including computer and software engineering, has been regarded as a way to solve many of the world's complex problems. Two examples are the Kiva project, where a website was created in order to help people struggling with poverty, and Project Masiluleke in South Africa, where mobile phones are now being used to help educate the population on HIV (Schwartz, 2013). In our research, the benefits of technology to solve social problems in the state of Victoria, Australia, were examined, leading to two studies assessing and enhancing the use of technology by a Banksia Garden Community Services (BGCS), an organisation that runs a community centre in the disadvantaged community of Broadmeadows. The first study involves the creation of a website for a youth group promoting advocacy for gender equality, and the second looks into the creation and use of a computer science curriculum for an afterschool study group. Previous successes in using technology to address social issues around the world were our broader context that served as a starting point for the implementation of technology in our research.

1.1: Using Technology to Address Social Issues

The two studies that we conducted involve utilizing available technology to supplement existing programmes at BGCS. Specifically, we chose two programmes due to the growing importance of the issues of violence against women and lack of education for younger residents in Broadmeadows. The limited use of technology in these programmes created the opportunity to study how the integration of simple technologies can advance these programmes and their missions. In the following two sections we provide general context of our targeted programmes.

1.1.1: Using Technology to Promote Advocacy

In 2002 several online programmes were studied to determine the most effective ways in which information is conveyed to youths. Individuals aged 15 – 24 years old were selected from a large number of youth workers from both Canadian and global programmes, and partook in a set of in-depth interviews about the use of information and communication technologies (Lombardo, Zakus, & Skinner, 2002). The results of this study showed that the use of technology has many advantages in spreading information, and more specifically messages about advocacy. Participants felt that websites spreading messages, similar to but not necessarily to do with family violence, provided a feeling of solidarity and a sense that they were not alone in their efforts to promote advocacy. They also felt that after receiving this information, they were empowered and felt that they could not just keep the information to themselves. Instead, the participants felt an obligation to pass the information on to someone else (Lombardo, Zakus, & Skinner, 2002). This shows that online tools for promoting advocacy are successful and can potentially not only reach users of the online source, but also others that receive the message second hand from one of the primary users.

An example of a successful online advocacy project is the “It Gets Better Project.” It Gets Better is a trademarked organisation that raises awareness and provides support to youth that are bullied or harassed for being a part of the LGBT (Lesbian, Gay, Bisexual, or Transgendered) community. The organisation uses its website to allow visitors to be educated through a series of powerful videos, facts, and stories that highlight the struggles that one faces growing up as an individual who is routinely bullied. The website also allows visitors to take an online pledge to stand up to inappropriate behaviour, and it has had over 596,000 people take the pledge and join their movement (It Gets Better Project, 2014).

In 2010, an organisation in the United States conducted a study on the effectiveness of online alcohol education programmes for college students, and similar results were produced.

This study had over 2,000 randomly selected participants split into both a control group and an “intervention group” (Croom, et al., 2010). The online learning programme included videos and follow-up quizzes after each video to reinforce the topics and lessons portrayed in the preceding video. The study found that the group who went through an online education course in alcohol and related risks performed significantly better on a post study test than the group who did not receive the online course. The group in which participants took the online course scored an average of 5% higher than the group in which participants did not receive any additional education on the topic (Croom, et al., 2010). The results of this study clearly depict the effectiveness of online education, including videos and quizzes, on the education of participants.

Online resources to promote advocacy have also become available in Spain where the issue of violence against women has recently received significant attention (Puente, 2011). There exist numerous online resources that bring this issue into the public eye. A study on the effectiveness of these websites has shown that utilizing technological resources is extremely effective in spreading a message. It was found that websites regarding violence against women not only have increased online reach, but that the message also spreads and strengthens offline efforts as well. The top four websites in Spain relating to advocacy of violence against women have over 1400 daily visits each, while the most visited site Fundación Mujeres has over 8 million entries and has a weekly bulletin with over 3500 subscribers (Puente, 2011). The number of people visiting these websites daily suggests that online platforms can help spreading messages on a national, even global scale (Puente, 2011). Online advocacy to raise awareness about violence against women has made an impact in Spain, and the techniques used to spread this message can potentially be extended to Australia.

These examples show that technology is currently used by many organisations as a tool to aid their efforts in numerous countries all over the world. It has become a tool that many organisations take advantage of in order to improve their effectiveness. However, the use of technology is not just limited to promoting advocacy; it can also be used to aid in supplementing the education of individuals with a wide range of backgrounds.

1.1.2: Using Technology to Improve Computer Education

The skills that computer science provides can help enhance future opportunities for individuals who do not possess even a basic knowledge of this technical field (Kafai, Peppler, & Chiu, 2007). There are three main methods of teaching computer science: object-first,

function-first, and state-first. In the object and function-first methods, students first learn about functions and objects respectively. This tends to be a more complex teaching method because it requires students to also understand all of the simpler programming components that are inherent for these more complex programming features (The Joint Task Force on Computing Curricula, 2013). In contrast, the state-first method has students writing code from day one and gradually introduces new concepts. In this method, the idea of programme states is emphasized, and only simple concepts such as variables and programme flow initially need to be learned. This has been widely accepted as the most effective approach to teaching students with no prior knowledge of computer science (The Joint Task Force on Computing Curricula, 2013). For individuals with limited existing computer knowledge, we therefore adopted a state-first approach when creating a computer science curriculum because it brings a more gradual flow from practice with more simple concepts to integrating complicated functions into complex scripts.

In the late nineties, one of the major obstacles facing the technical community was to bridge the so-called digital divide, where disadvantaged areas have significantly less access to computers and, thus, less computer literacy. The embodiment of this mission manifested in the Computer Clubhouses, a series of community technology centres in disadvantaged areas. They provided a place for youth to gather and learn to use many computer technologies, from word processors to photo, music, and video editing. However, programming was not initially included in these tasks (Kafai, Peppler, & Chiu, 2007).

This changed in 2008, when Scratch, a visual programming language and design software intended to teach programming concepts to children was introduced in a Computer Clubhouse in South Central Los Angeles (Lifelong Kindergarten Group, n.d.). This Clubhouse served mainly underrepresented people aged 8-18 from one of the city's most impoverished areas. Over the first two years after the addition of Scratch to their activities it grew to be the most widely used software. Very little was done to teach the clubhouse members programming concepts; instead they were allowed to work on their own projects and could request help from mentors when needed. The Clubhouse also organized 'Scratch-a-thons' where members would work on a project for 3-4 hours and then present the projects they had created. Over the first 18-months, over 536 projects were created in Scratch as opposed to other non-programming based computer programmes. Scratch accounted for 34% of all the projects created at the Clubhouse. A detailed analysis of the projects showed that many programming concepts can be taught, including basic concepts such as user interaction, loops and conditionals, and more advanced concepts such as synchronization and

randomization (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008). This case is of particular note as the members of the Clubhouse were not provided with any external help in learning Scratch. All undergraduate mentors who worked with the Clubhouse members had no experience with Scratch before its introduction. This led to a culture where a few members became the go to experts, and would often teach other members and their mentors what they had figured out on their own (Kafai, Peppler, & Chiu, 2007). This suggests that a similar programme could have success in Australia, and require minimal resource investment as Scratch can successfully be self-taught.

The use of technology in education, and more specifically computer education, has been on the rise, and its integration to supplement current education programmes has been successfully demonstrated in many areas. The success of these programmes along with programmes utilizing technology to promote advocacy was used to help the community organisation Banksia in its efforts to teach computer skills for the youth in Broadmeadows.

1.2: Family Violence in Australia

Australia, as a nation, faces a very real problem in regard to family violence - specifically violence against women. One in three Australian women has experienced physical violence since the age of 15, and nearly one in five has experienced sexual violence, as shown in Figure 1 (Domestic Violence Statistics, 2015). A survey conducted in 2005 showed that almost half a million Australian women had reported being either physically or sexually assaulted within the last year (Domestic Violence Statistics, 2015).



Figure 1: Rates of Violence against Women in Australia (Domestic Violence Statistics, 2015)

Family violence is present in every state of Australia, and Victoria is no exception. Victoria reported 60,829 incidents of family violence in 2012. This is an increase of 72.8% since 2004 (Key Statistics, 2014). Almost two out of five people who have reported these actions have said that it had been ongoing for over two years (Victims Support Agency, 2012). However, it is estimated that only a fraction of those who experience family violence report the abuse (Victims Support Agency, 2012). Thus, this indicates that there may be an underreporting in the number of actual incidents that occur. This is an issue because the number may not reflect just how serious of an issue family violence is within the Australian community.

Although the police do not provide public records on family violence for local government areas, there are figures suggesting that violent crime is exceptionally high in some areas. For instance, the city of Hume, a suburb of Melbourne, the capitol of Victoria, has very high crime, and more specifically rape rates. The incidents of rape in Hume increased by 44% between 2013 and 2014 according to police reports (Crime Statistics by LGA, 2014). This increase is the greatest increase for any crime against a person in Hume, with assault increasing the second most at 2.2% (Crime Statistics by LGA, 2014). These crime rates show that in addition to Victoria and all of Australia struggling with family violence, some smaller communities are faced with an even greater challenge.

1.3: Computer Education in Australia

Australia as a whole is among the top countries in the world when it comes to education rates, with 82% of people aged 15 to 19 years old enrolled in a formal study (Education and Work, Australia, 2014). However, there also exist certain areas of education in which Australia is struggling. The number of people studying information technology was cut in half over the past decade from 8.5% to 3.3%. Of the 2.9 million people enrolled in formal study, only just over 500,000 were enrolled in technical and further education institutions according to a 2014 census (Education and Work, Australia, 2014). In addition to these gaps in technical education throughout Australia, the state of Victoria also struggles with some aspects of schooling.

In Victoria, the total percentage of citizens enrolled in education is similar to that of Australia as a whole, but when one delves deeper into the statistics, small gaps appear. The percent of students in Victoria enrolled in non-private primary schools is 16.8% compared to 18.2% of Australia. In addition, in the five years following a 2006 census the percent of students enrolled in non-private secondary schools dropped by 0.6% (2011 Census Quick Stats, 2013). Although these numbers may seem insignificant, a 1% change in the education statistics of Australia corresponds to almost 30,000 students.

In addition to the struggles with education in Victoria and Australia as a whole, there exist communities that have fallen even further behind in respect to this issue. One such community is Broadmeadows, which is located in Hume, Victoria. The Broadmeadows community is very rich in cultural diversity but is devastated with unemployment, poverty, illness, and lack of higher education (Australian Bureau of Statistics, 2014). The community consists largely of immigrant groups: 16.1% immigrated from North Africa and the Middle East, and over half of the Broadmeadows' population was born outside of Australia (Australian Bureau of Statistics, 2014). In addition, residents of this community are less likely to have access to higher education as evidenced by very low rates of individuals pursuing university or tertiary education (Australian Bureau of Statistics, 2014). As seen in Figure 2, the year 12 graduation rates for Australia and Victoria are quite good, at 74% and 88.1% respectively (2011 Census Quick Stats, 2013). Melbourne and Broadmeadows are in a different situation, with graduation rates of 55% and 35% respectively (Support Banksia Gardens Community Services, n.d.). Education is an essential tool in carrying a community out of poverty and it is the goal of Banksia Gardens Community Centre to increase the graduation rates in the Broadmeadows area. Another issue is that while residents have access

to technology, many of them do not have in-depth or advanced computer knowledge (Australian Bureau of Statistics, 2014).

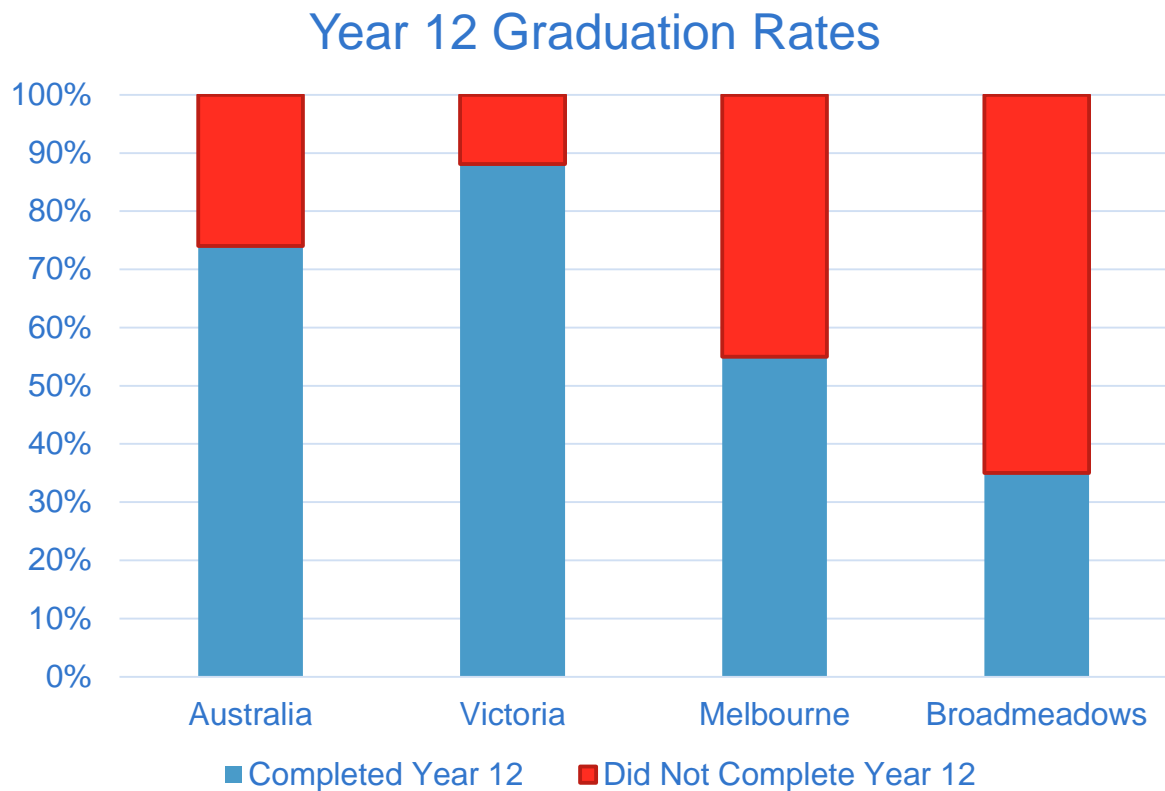


Figure 2: Year 12 Graduation Rates in Australia

1.4: An Ideal Study Location: Banksia Gardens Community Services

Broadmeadows is the prime location to study the effective use of technology in an attempt to solve a range of issues facing the community. As a disadvantaged community, there exist numerous public organisations focused on improving the situation in Broadmeadows. One of these organisations, Banksia Gardens Community Services (BGCS), is a charitable not-for-profit community service centre dedicated to supporting and providing opportunities to the community. These opportunities range from a work skills programme intended to help community members improve their job prospects, to technology courses, and an after-school computer club for the youth of the area.

BGCS also works closely with and hosts the Good People Act Now (GPAN) Youth Group, as one of many partner organisations. GPAN was created as an initiative within BGCS to educate Broadmeadows' youth about family violence and violence against women (Good People Act Now (GPAN), 2013). Due to the overwhelming statistics that demonstrate the problem of domestic violence and violence against women, many groups have formed in order to raise awareness. GPAN is focused on women's advocacy with its attention directed towards Broadmeadows. The group holds programmes and events that teach young children in the community the proper relationship practices along with bystander action presentations that inform participants about safe and effective ways to intervene if they see a situation that could lead to violence. An important goal for GPAN is to bring awareness of this major issue to the community because many Australian citizens are unaware at how common domestic violence is. This could be a result of the fact that nearly 65% of women who have been physically abused do not report it to the police, and over 80% that have been sexually abused do not report it (Domestic Violence Statistics, 2015).

GPAN is working to educate young people about the seriousness of family violence, features that make up healthy relationships and the proper places to look for help escaping an unhealthy relationship. They also provide educational information on how to intervene when witnessing family violence, which is called bystander action. However, their impact is limited by the group's small size of approximately 15 trained members. Technology can be used to assist their mission, as technological resources can greatly enhance the reach of even the smallest organisation (Puente, 2011). Thus, GPAN is hoping that an online advocacy project will help it raise awareness of the issues associated with family violence and provide a tool to help individuals address family violence.

BGCS has also made efforts to improve the community of Broadmeadows and address the lack of computer literacy through community education programmes. The Community Centre has organized an afterschool study group aimed at high school aged children, providing them opportunities to supplement their education, as well as increase opportunities for learning about subjects not necessarily focused on in school. This programme has the opportunity to educate students on computer-related topics, such as computer science and programming. The creation of a computer science curriculum could provide additional opportunities for the youth involved in the club.

With the growth of computers and technology as a whole, the employment opportunities for those with a formal education in computer-related fields have become much more desirable. Not only has the computer industry continued to grow over the last few decades, but jobs in other fields are beginning to require more and more computer knowledge (Lawson & de Matos, 2000). Therefore the ability to provide young students with computer knowledge will enhance their chances of earning a degree and more importantly a career. The second aspect of this study is aimed at the continuation of this mission through the design of a computer science curriculum for BGCS.

It is clear that BGCS faces many challenges in the community of Broadmeadows, ranging from social to educational issues. Even though BGCS has limited resources, the use of technology presents an opportunity to enhance its impact on the community. This was explored through the two following studies involving the creation of a website for GPAN in order to promote awareness of family violence, as well as through the creation of a computer science curriculum for BGCS.

Chapter 2: Study 1

The GPAN youth group is a relatively new group dedicated to raising awareness about family violence and violence against women. The group consists of 15 volunteers who would like to extend its reach to the community of Broadmeadows through the integration of technology in the organisation. Thus, the goal of Study 1 was to study how technology could help GPAN spread their message through the collaborative development and implementation of an online advocacy programme for the youth group. To do this, we conducted focus group sessions with GPAN to better understand the organisation, its goals, and to discuss content that would be appropriate for a webpage, as well as discussing ideas for interactive on-line videos. Once the website and videos were created, we obtained user feedback on the website's content and usability as well as the usability of the interactive videos.

Methodology

Objective 1: Collaboratively develop and implement an online advocacy programme for the Good People Act Now Youth Group

Focus Groups with GPAN Members

Participants

GPAN consists of 15 trained members who use connections in the community to help run events. Twelve trained GPAN members (66% female) participated in the focus group. The mean age of these participants was 19.

Procedure

To better understand GPAN's goals and objectives, we conducted focus groups consisting of the active members of the group. To conduct the first focus group, we attended one of GPAN's weekly meetings. After everyone agreed to participate and introduced themselves, we began the focus group. We first asked questions to better understand GPAN and how it operates. We asked about the day-to-day operations and its current resources. For example, we asked about the time and resource investment that went into putting on an event at a festival the previous month. We also asked the GPAN members to discuss their vision for the future of the organisation. In addition, we were particularly interested in GPAN's opinions on the creation of its website. We assessed the audience they wanted to target, the aesthetics of the website, the specific content to be included in the website and the maintenance needs and requirements. For example, to learn more about the projected audience, we asked GPAN to explain the audience they envisioned viewing the website. In an

effort to assess the aesthetic preferences of GPAN we also asked about the themes and colours that they believed would best fit with any existing GPAN materials. Furthermore, we asked a series of questions regarding the content for the website. More specifically, we enquired about the multimedia resources they would like to see on the webpage, about the most important information that needed to be on the webpage, and where on the webpage they wanted different content to be located. For example, we asked where on the website they would like their contact information, and what information they were comfortable with being published online. Finally, we asked GPAN members questions that allowed us to better understand their needs and goals for maintaining and updating the webpage in the future. We asked them how often they envisioned needing to update the webpage, and how many people would need the appropriate administrative privileges to make these updates. We also asked GPAN if anyone had any web development or administration experience and if anyone was interested in maintaining the website in the future. After completing the focus group, we thanked all the GPAN members for their participation. See Appendix B for a list of questions that were covered during the focus groups.

To conduct the follow-up focus groups with GPAN, we attended four additional meetings. After any member new to the focus groups had agreed to participate, we presented the current state of the website, with a focus on any changes made since the last meeting. We then asked the GPAN members about their opinions of the changes, and for any suggestions to improve usability or flow. In these focus groups, we also discussed ideas for the implementation and filming of the on-line interactive videos that are used as part of the Bystander Action Tool. The Bystander Action Tool is a series of online videos that present every-day scenarios in which viewers are educated on the actions they can take when witnessing inappropriate behaviour in regards to gender inequality. The viewer is educated on how to challenge negative views and actions throughout the videos with questions and information blocks that pop up at specific points during the videos.

Framework for a GPAN Website

In order to study the effective use of technology with GPAN, we used WordPress as a platform for the development of a website. WordPress is free open source web development software and content management system, already used as the host for Banksia's website. We were able to use the live web editor to organize content on the site and create all needed pages. During the development process the page was locally hosted off a laptop that allowed

for multiple editors on the same wireless network. This allowed for the site to remain offline until its completion and publishing by BGCS.

Creating Bystander Action Videos

The bystander action videos address issues such as respectful and healthy relationships, as well as taking action when confronted with viewpoints that lead to unhealthy relationships. Four videos, each of approximately 30 seconds in length, were created with a target audience of the youth between the ages of 13 to 16. The scenarios and scripts for the videos were developed, edited, and acted out in collaboration with BGCS and GPAN members through the focus groups. The topics for the four videos are as follows:

- Break Up Scenario – Challenging stereotypes about masculinity
- Sports Club Sexism – Challenging sexist and violent jokes
- Chris Won't Leave Mads Alone – Responding to persistent sexual advances
- The Jealous Boyfriend – Recognizing unhealthy and controlling behaviour in relationships

We assisted in the coordination of the filming process, but the organisational and logistical aspects of the videos during all stages of development were completed by an outside filming and editing company.

On the day of filming, all GPAN members, and soon to be actors, arrived an hour prior to the arrival of the filming company in order to rehearse and perfect the scenarios. The filming was all done on location at Banksia Gardens Community Centre in many of the different, available rooms in addition to multiple outside locations. Each 30 second scenario took an average of 45 minutes to record with about 10 minutes set up at each location. This resulted in four hours of filming, all taking place on one day. The editing was completed in-house at the filming company's studio and after two days of editing.

In addition, we made the edited versions of the videos interactive in nature. Specifically, multiple-choice questions near the end of the videos present several ways to respond to the enacted situations. The viewer selects an option, and is presented with statistics on the percentage of viewers that selected each response, as well as a suggestion for a positive response.

We implemented the interactivity using H5P, an HTML5-based plugin for WordPress. This plugin was designed for integration of interactive media with webpages and provides support for multiple choice questions, along with many other question formats. In addition to

including the necessary features to make the bystander action videos a reality, H5P also has a graphical user interface for the addition of questions to a video, meaning that GPAN members are able to add additional videos in the future without any web development knowledge.

Evaluating Usability and Appeal of GPAN Website

Participants

To further improve the GPAN website, we performed a usability study. In particular, we conducted a study consisting of 22 participants (9 males; 13 females). The participants consisted of local citizens visiting Banksia Gardens Community Centre, GPAN members and Banksia staff, and the mean age of the participants was 34. The wide range of participants helped us to receive the viewpoints of a broad spectrum of different demographics. This assisted us in identifying the key features and content of the website that are found useful by a wide range of individuals.

Procedure

To conduct the surveys, we utilized the local community centre where we were given permission to survey visitors. We were able to set up near the entrance of the centre and ask individuals visiting the community centre to look at the webpage and give us their feedback. The survey (See Appendix C) was geared toward the functionality and ease of use of the website as well as the effectiveness of the Bystander Action Tool. The survey asked about the participants' interactions with the website in its entirety as well their experience with each independent page. Questions included rating the usability of each page of the website on a 7-point Likert-Type scale (1 = Very Difficult; 7 = Very Easy). Another question referred to how informative the Bystander Action Tool was (1 = Not at All; 7 = Very Much). Lastly we asked for comments for any additional feedback or suggestions to do with the usability or content of the website. These surveys helped us with the need to understand the ease of navigation from page to page as well as the clarity of purpose of each page.

Deliverables and Outcomes

Developing a GPAN Website

The content and setup of the webpage was a result of topics discussed in the focus groups with GPAN members. The website contains a newsfeed to display information on recent and upcoming GPAN activities, as well as relevant local news. The site also includes a page with links to family violence support services, and a page containing the interactive

bystander action videos. Additionally, to make the website easy to maintain for any public organisation, it was made customizable without web development knowledge.

The Bystander Action Tool, due to the interactive and visual nature of the included videos, is the youth group's method of choice to reach out to its target community. The videos replicate everyday scenarios in which inappropriate action or behaviour takes place. The website design was influenced through both the focus group of GPAN members, as well as through the survey of community members. The content was workshopped through five different focus groups with the GPAN members.

Through the creation of a website for GPAN (Appendix A), we were able to examine the implementation of technology into the organisation. The website we developed provided the organisation with a hub for its online outreach, as well as the Bystander Action Tool, a resource to use in its offline training.

Set Up of Webpage

The website is designed to help GPAN not only broadcast its mission to the community of Broadmeadows, but also make the information and Bystander Action Tool available to anyone across the world with access to the internet. The group's mission is to educate the youth in the area about the issue of violence against women and promote gender equality as well as healthy relationships. Ideally, GPAN hopes to reach people and educate them on signs to look for in relationships before abuse begins.

The goal of the website and what important features are needed to convey the group's message were discussed in detail and at great length during each focus group meeting. One of the common themes of each focus group meeting was that the website needed to be easy to use and the messages it promoted had to be simple, yet effective. The website was geared toward children of ages 13 to 16 years old and therefore it was attempted to be set up in a teen-friendly manner; consequently, and similar to other websites intended for children of this age group, it was composed of a majority of pictures and videos as opposed to lengthy text sections. The website now consists of the homepage and six tabs for each individual page, including: Get Smart, Get Help, Get Acting, Get Along, Bystander Action Tool, and Contact Us. The homepage, shown in Figure 3, consists of a visually appealing slider that hosts an introduction video in addition to pictures of local community members pledging to take a stand against family violence. At the top of the homepage is the standard header of the website that hosts the GPAN logo as well as the logo for BGCS which serves as a link to

Banksia's website. There is also a small logo of the sponsoring organisations of GPAN and the State Government Victoria.



Figure 3: Homepage of the newly created GPAN website

The website is designed to help GPAN not only broadcast its mission to the community of Broadmeadows, but also make the information and Bystander Action Tool available to anyone across the world with access to the internet. The group's mission is to educate the youth in the area about the issue of violence against women and promote gender equality as well as healthy relationships. Ideally, GPAN hopes to reach people and educate them on signs to look for in relationships before abuse begins.

The Get Smart page of the website, shown in Figure 4, informs readers of the issue of violence against women and shows the prevalence of such violence in the area. Similarly to the homepage, this page hosts a slider, but with much different content. This content includes videos that inform viewers about the issues with family violence, as well as some resources viewers can use to educate themselves on the issue. Located directly next to the slider is a summary of some of the important local statistic relating to family violence and violence against women. This local information allows the website to reflect the area that the group is centred around and the local nature of the youth group. Additionally at the bottom of the page

are links to other sites where visitors can find more information relating to the issues of family violence and violence against women.



Figure 4: Get Smart Page

GPAN also aims at connecting victims of violence and individuals who have been exposed to violence with organisations that have expertise in the particular area that an individual is struggling with. The Get Help tab of the website, shown in Figure 5, allows the user to be connected with specific organisations in order to receive help from an appropriate source, if they have experienced any form of family violence. The different organisations presented on this page each have a specific demographic in which they specialize their focused aid. Each organisations logo at the bottom of the page acts as a link to that organisation's website and help service.

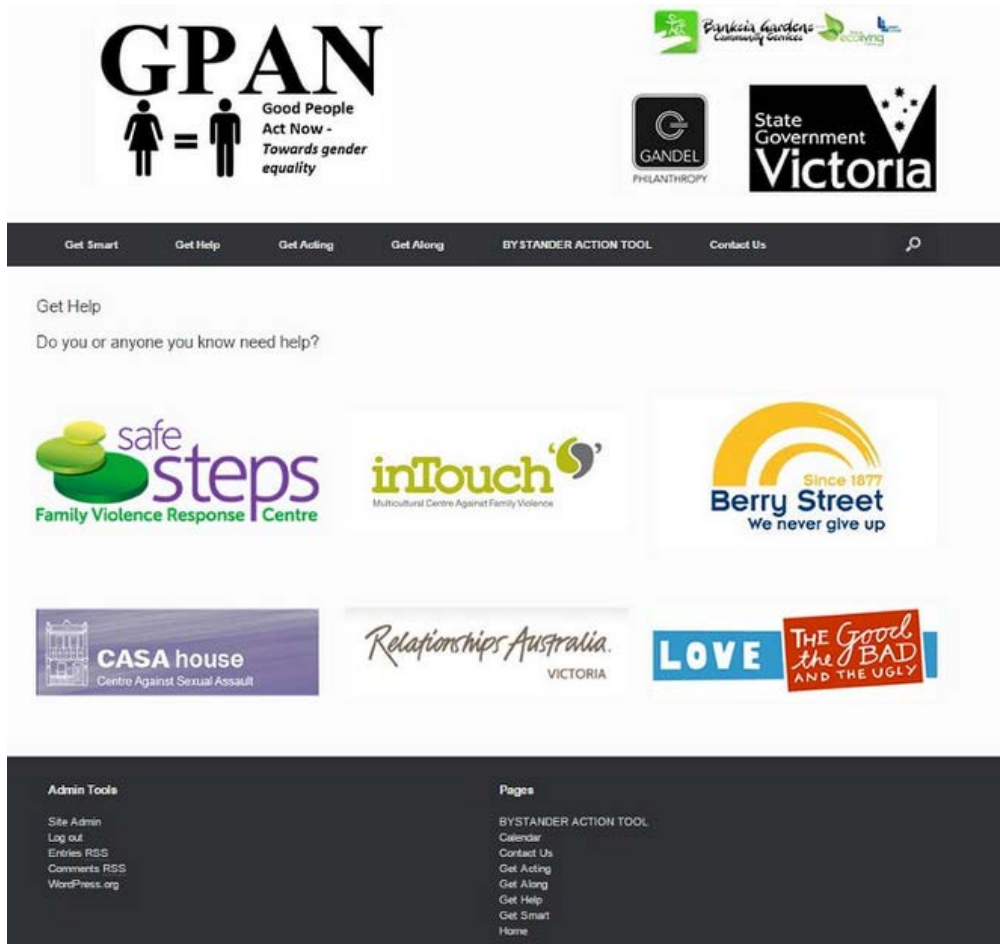


Figure 5: Get Help Page

The Get Acting tab informs the visitor of appropriate actions to take in an effort to prevent family violence. Paralleling a goal of GPAN, as discussed in the focus group meetings, this page contains videos and links to relating sites that not only discuss how to take action against violence, but also give examples of how one could react to different situations appropriately. The layout of the Get Acting page is similar to the Get Smart page and can be seen in Figure 6.



Figure 6: Get Acting Page

Another desire of the group that was clearly discussed during the focus group meetings was that GPAN would like to ensure that the users of the website have the opportunity to actively participate in the events that the group hosts and know how to get involved with GPAN. The Get Along tab, seen in Figure 7, consists of a calendar of upcoming events with details relating to each event and a bulletin board for the members of GPAN to advertise future and past events. The board supports easy to use visual and text interfaces so that with a short provided instruction manual any member of GPAN has the ability to post a story about an event with pictures or even videos.



Get Along

Craigieburn Festival

At the festival we set up our tent and encouraged festival goers to take a pledge to stand up against violence



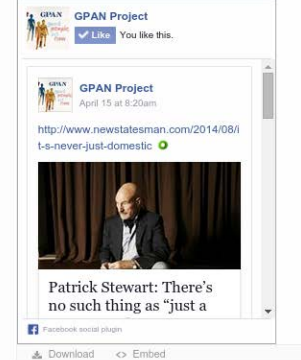
Upcoming Events

There are no upcoming events.

Add -

View Calendar →

Find us on Facebook



Archives

Figure 7: Get Along Page

The final and maybe most important tab is the Bystander Action Tool. The tool, shown in Figure 8, consists of four interactive videos that were produced by a professional film crew. Over the course of a day, GPAN members and our team directed and acted in the scenes. The film crew was then able to edit the videos to a final version that show the user everyday scenarios of violence against women. Our team was able to implement the interactive components of the videos that demonstrate the proper responses and actions in those scenarios.



Figure 8: Bystander Action Tool Page

The Bystander Action Tool was a product of many discussions and focus group meetings with GPAN. Our Bystander Action Tool gave GPAN a tool that allows them to make strides towards accomplishing its goal of reducing violence against women. Specifically, the tool teaches users the skills involved in recognizing violence and inappropriate actions, and perhaps most importantly, how one should respond in certain situations. Through the focus group meetings with GPAN, it was decided to create four interactive videos that would show four examples of typical scenarios and proper responses. Each scenario depicted in the videos was intended to emphasise how an individual should react in response to either witnessing violence or being a victim of violence. The titles and themes of each scenario are outlined in Table 1.

Table 1: Bystander Action Tool Scenarios and Themes

Title	Theme
Break Up Scenario	Challenging stereotypes about masculinity
Sports Club Sexism	Challenging sexist and violent jokes
Chris Won't Leave Mads Alone	Responding to persistent sexual advances
The Jealous Boyfriend	Recognizing unhealthy and controlling behaviour in relationships

Each one of these scenarios has various questions integrated into the flow of the video that challenge the user to think about possible appropriate actions an individual should take in situations that may be similar to the videos. For example, after a character calls a classmate a “homo,” a question appears that asks what stereotypes this comment reflects and how one can challenge these stereotypes and support the classmate being made fun of. The video then continues after the user answers provides the correct response, and depicts how that response may play out in every-day conversation. Once all of the content was integrated into the website, both the website and the Bystander Action Tool were analysed in a user survey as described in the following section.

Webpage User Survey

The 22 user surveys of the website allowed us to examine how local community members perceived the site and whether or not they thought the Bystander Action Tool would change how they act when witnessing any form of violence or action reinforcing gender stereotypes. When asked about the probability that they would act differently, on a scale from 1 to 7, or not likely (1) to very likely (7), the mean response was 5.7. This shows that for the sample of community members surveyed, the Bystander Action Tool allowed people to report that they are likely to act differently in the future if a similar situation were to arise.

When asked to rate the ease of use of each page of the site, including the Bystander Action Tool, participants were again given a scale from 1 to 7, this time representing a scale from impossible to use (1) to very easy to use (7) . The mean response was 5.8, indicating that users were easily able to navigate the website. The page with the lowest ranking was the Get Along page with a mean of 5.4 and this value showed us that we needed to make changes to how that page was set up. We made the navigation around this page correspond more closely to that of the home page because the homepage navigation was rated as the easiest to use by visitors with an average ease of use value of 6.1.

Survey participants were also given the option to leave comments about the web page and how it could be improved. We were able to take these comments about particular links or smaller pieces of content into consideration and make adjustments to these finer details of the website so that it corresponded with the needs and wants of the local community members.

We took one comment very seriously, as it mentioned that this survey participant was a family violence survivor. The women's comment was "Although the webpage is geared towards women, there should also be something about making it clearer that this is not just a women's issue." The comment resulted in an additional video titled #MenLetsTalk which was created by an organisation targeted to initiate males talking about the issue of violence against women. Content was added to the website that shows how family violence is a male issue as well, and getting men to talk about the issue is a step in the right direction.

Discussion

Through focus groups and working alongside GPAN we were able to determine an opportunity to study the use of technology in a programme run by a social organisation. To complete this study we created a webpage for the youth group with the aim to aid them in promoting advocacy for the issue of violence against women. Before the website was created, GPAN had been a group that relied on word of mouth and various hosted awareness events. Focus groups with GPAN led to the conclusion that a website would be an ideal use of technology for the organisation and would allow us to examine how this technology could help group pursue its mission.

In addition to the website, the focus group meetings lead to the idea of the website hosting a new tool that could also be developed: the Bystander Action Tool. This set of interactive videos shows scenarios that depict scenes involving gender inequality and provides viewers with information on how one could appropriately respond. These two pieces of technology, the website and the Bystander Action Tool, were successfully integrated into the operation of GPAN; it serves as examples of how technology can successfully be deployed by community organisations.

Through the Bystander Action Tool and supporting website, as discussed in the focus group meetings, the GPAN group feels better equipped to continue its mission to challenge gender stereotypes and educate people on the issues of violence against women. The fact that the website was created in WordPress makes it easy to use, even by individuals with limited computer knowledge. The Bystander Action Tool helps inform the viewers on the issues relating to gender inequality.

Although websites are not a new technology, GPAN will now be able to expand its online presence and thus spread its message. When the site is published online, it will be important for the group to promote the site on search engines as well as on social media in order to ensure maximum page views. We suggest that the group updates the webpage and supporting software regularly and provide a calendar for keeping track of updates. In addition to regularly updating the software supporting the site, we recommend using the site in conjunction with the group's existing Facebook page to inform local community members of upcoming events as well as publishing updates of past successful events.

The Bystander Action Tool will also allow for future possibilities to improve the outreach and message that the group is promoting. GPAN has already heard of schools in the local and broader Australian community which have expressed interest in the integration of the Bystander Action Tool and GPAN group into the important life lessons that they teach students. In summary, the website and Bystander Action Tool are two pieces of technology that could increase potential for GPAN to grow. Local schools that GPAN has spoken with have already expressed interest in the website and tool, and these connections can lead to a greater impact in the future.

Chapter 3: Study 2

Banksia operates many programmes for the local community in order to further the education of the citizens of Broadmeadows and better equip them with career tools. This study was designed with the key idea of using technological resources to help Banksia Gardens Community Services educate the youth of Broadmeadows in the field of computer science. By creating a computer science curriculum, we will be able to study the use of technology in another programme run by BGCS.

Methodology

Objective 1: Analyse the amount of computer science knowledge of the staff and patrons of Banksia

Interview with Banksia Gardens Staff

Participants

We conducted five interviews with key staff members of the BGCS team. The staff members interviewed, included the CEO, the Deputy CEO and Manager of Community Development and Action Research, the Community Development Projects Coordinator, the Project Coordinator for Aiming High VCE Support, and the Computer Club Facilitator. These staff members were chosen due to their relations to the programmes involved in this study. We also interviewed the Hume Technology Guide, who works closely with BGCS to provide technology resources to the Centre.

Procedure

By using the style of semi-structured interviews, our conversations with the Banksia staff members possessed a more natural flow, which resulted in an understanding of each interviewee's position on not only the interview questions, but also other relevant topics that arose throughout the interview (Barriball & While, 1994). This conversational style generated different responses from each member that were explored, and the interview style created a more complete image of the challenges faced by the community centre's staff.

The interviews were conducted to gain an understanding of exactly what computer resources were available. It allowed us to determine what resources and software were currently available at Banksia and what we were able to use in the design of a programme that highlights and utilizes those resources. During the interviews, we made sure to cover specific topics, including the interviewee's knowledge of the resources available as well as what resources would be useful to have. For example, we asked what computer programs they were familiar with that are currently in use in the organisation's computer club. Another

concern we had to address in the interviews was the viability of implementing a simple computer programming curriculum in the computer club. We asked staff involved in this programme about the attention spans and capabilities of the computer club students. In addition we also inquired about other programmes that Banksia operates that could be investigated as alternatives to the computer club for the implementation of our curriculum. It was important to get an idea of all of the relevant initiatives run by Banksia so that our curriculum could be implemented into the appropriate programme. The programme that the curriculum was ultimately placed under and customized for was decided through these interviews with the staff members.

Objective 2: Develop and trial a self-directed computer science curriculum for Banksia Gardens Community Centre

Creation of a Computer Science Curriculum

A curriculum is important in providing structure to individuals looking to increase their knowledge in the highly technical field of computer science (Cooper, Dann, & Pausch, 2003). Of the numerous possible programming languages that can be used to teach computer science, we made the decision to use Scratch for several reasons. First, Scratch is a free, open-source visual block-based programming language developed by MIT, and available at <https://scratch.mit.edu/>. It is available in both an offline downloadable version and an online browser-based environment. The software has also seen extensive use and success in use at community technology centres all over the world ranging from a centre in Caracas, Venezuela, to multiple centres in Cambridge, Massachusetts, USA (Magrane, 2015). The effectiveness of teaching programming concepts to children has been well documented and it was shown that Scratch can be learned easily without additional help from staff who already know the language (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008). The combination of free software, availability to the students outside of BGCS, and the success it has had with other community centres made it an ideal choice for the curriculum.

We created a series of modules, each based on a group of concepts, determined to be level appropriate, through observations and one on one interaction with the students. The creation of the modules incorporated feedback from Banksia staff which was obtained during the semi-structured interviews as described in Objective 1. Based on the observed computer science knowledge of the target audience, each module contains a tutorial with pictures of the relevant commands in Scratch, and takes users through the process of creating a complex project. At the end of each module there is a prompt for an individual project that will

challenge users to go above and beyond their previous programming knowledge, while utilizing the methods taught in that individual module.

Surveys of Aiming High Students

Participants

The students surveyed in this aspect of the study consisted of 15 (7 male, 8 female) Aiming High participants with an average age of 17. These students were chosen due to their commitment to academics, made apparent in interviews with Banksia staff members and the mere fact that they are attending afterschool sessions to improve their knowledge.

Procedure

Before the students were given the opportunity to complete the computer science modules, they were instructed to complete a pre-lesson survey that analysed the participants' prior computer science knowledge as well as their level of interest in computer science. The survey asked participants on a 7-point Likert-Type scale (1 = Very unfamiliar; 7 = Very familiar). These concepts include variables, Boolean logic, user interaction, lists and concurrency. The full pre-Module survey can be found in Appendix E.

Following each module, the students who finished it were instructed to complete another survey. This post-Module survey asked about the students new understanding of the same computer science concepts asked about in the pre-Module survey. It also asked for feedback on the module itself – how engaging it was, and the module's level of difficulty as well as demographic information of the participants. After the survey and analysis, the module was updated and revised based on the results in order to make it cater more to the students' needs.

Deliverables and Outcomes

Understanding the level of computer science knowledge of the Banksia staff was important to us as we began developing our computer science curriculum. Our interviews with different staff members resulted in a few key pieces of information. First, we discovered that only a few staff members have used platforms similar to Scratch, such as Ruby and Python, but have not used Scratch in particular. From the interviews with the staff we decided to look into other after school programmes that Banksia offers to implement the computer science curriculum we had developed. Originally we had decided to implement the curriculum in the computer club programme, but after the interviews with BGCS staff, we looked into the Aiming High programme. Aiming High is a programme Banksia hosts for young adults ranging from 15 to 18 years of age who are looking to continue and improve their education

outside of school hours. Aiming High is composed of roughly 30 determined teenagers who showed interest in learning new concepts and were decided to be the proper fit for a new computer science curriculum. . The observations of the group combined with suggestions by the staff members lead our group to implement our computer science curriculum into the Aiming High programme with relative success as described in the following section.

Computer Science Curriculum

Scratch is a tool that helps teach basic computer science programming skills. It was designed to be used with a younger age group with the idea that it is easy to use but engages the participant. With just a few easy commands the user can have their characters moving around. The curriculum consists of three modules; Introduction to Scratch which uses the “Getting Started with Scratch” guide developed by MIT, Module 1: Pong, and Module 2: Space Invaders (Scratch Project Editor, 2015). The modules are focused in the areas outlined in Table 2, which were determined to be optimal based on the results of Objective 1.

Table 2: Topics and concepts covered by each module

Module Title	Topics and Concepts Covered
Introduction to Scratch	<ul style="list-style-type: none"> • How to access Scratch • Pre-existing “Getting Started with Scratch” guide
Module 1: Pong	<ul style="list-style-type: none"> • Storing and manipulating variables • Boolean operators, conditionals, and loops • User Interaction and scripting
Module 2: Space Invaders	<ul style="list-style-type: none"> • Lists • Concurrency • Inter-process communication • Recursion • Program State

The Introduction to Scratch is a very basic module that teaches the user how to open the program and navigate to Scratch’s own introduction. The introduction teaches the user the basics of Scratch by incorporating a simple sprite following basic commands. The users are taught how easy it is to make sprites move with a limited number of simple commands. The final product of the “Getting Started with Scratch” guide is a program that results in the sprite dancing to music that the user incorporates through programming. To accomplish this program, the student is given a step by step instruction manual in order to complete the module. A screenshot of the final product can be seen in Figure 9.



Figure 9: Program created by following “Getting Started with Scratch” guide

The pong module is the first module of the curriculum that integrates and explains more complex programming concepts. The module takes students through creating a version of the classic arcade game “Pong” where a ball bounces back and forth between two player’s paddles. The goal of the game is to use a paddle to bounce the ball past an opponent’s paddle. Through the creation of this game, students are introduced to basic concepts including Boolean operators as well as while loops and user interaction. The students are again given a step by step instruction manual, specifically created for this curriculum that instructs them how to code this game; it introduces the programming concepts used through simple explanations. By following the instructions students end up with a final product shown in Figure 10. Following the module is an optional bonus work in which students are encouraged to use the concepts that they learn to integrate additional features into the game, such as using the space bar to start the game, a score board, and a game over screen displaying the winning player.

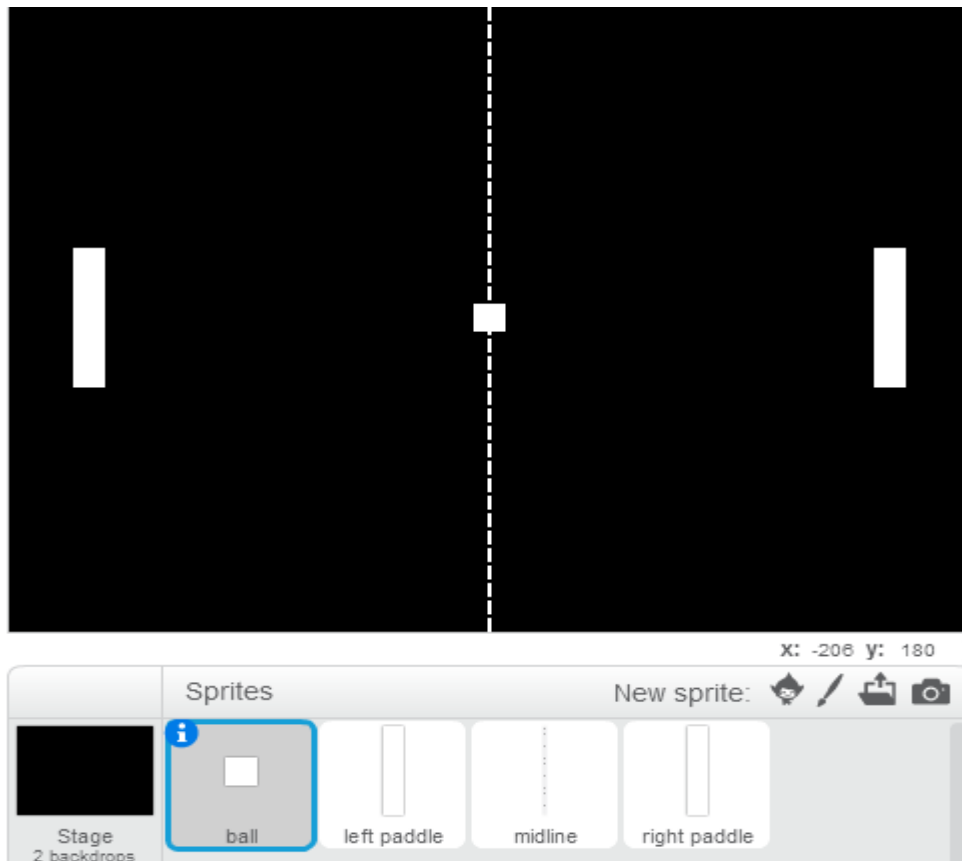


Figure 10: Pong game created by following Module 1 instructions

The second module takes the participants through the creation of a version of space invaders. Space invaders is another classic arcade game where a player has control of a space craft and must destroy alien ships while avoiding getting hit by alien projectiles. This second module introduces different commands that are longer and more complex than the code used in the introduction or first module. Concepts such as lists, concurrency, program state and others as outlined in Table 2. The curriculum uses the Scratch program along with our step by step instructions in order to create each game. The idea is that these modules are virtually self-taught so that the users can go through the modules on their own or without the help of an instructor or trainer.

The step by step instructions used in each module are designed based on the assumption that students have no previous programming knowledge or experience. For the “Introduction to Scratch” guide and the beginning of Module 1, students are taken line by line through the writing of the code and what the addition of each line does in the code. As Module 1 advances and throughout Module 2, the line by line instruction transitions into a block by block teaching method, where blocks of code are investigated, and the code’s interaction within each block and the program are discussed. At the end of each module are

suggestions for further work on the games or places where the students can go to continue their education on Scratch and programming in general.

After the modules were completed, the first module was tested by making it available to participants of the Aiming High programme. The second module was not tested due to time constraints. Of the students surveyed, none had any previous programming experience and only one had a prior interest in computer science. This parallels surveys conducted in American high schools where about one quarter of students were interested in and completed a programming course (Is Coding the New Second Language?, 2014). The students who completed the module showed an increase in familiarity with the concepts taught in Module 1, consisting of variables, Boolean logic, loops, and conditionals. The results of our survey can be seen in Figure 11. Students reported an increased interest in programming after completing the module with the mean interest level increasing from 4.375 to 5 from the pre to post surveys. This shows that the Module has stimulated an interest in the topic among the students surveyed.

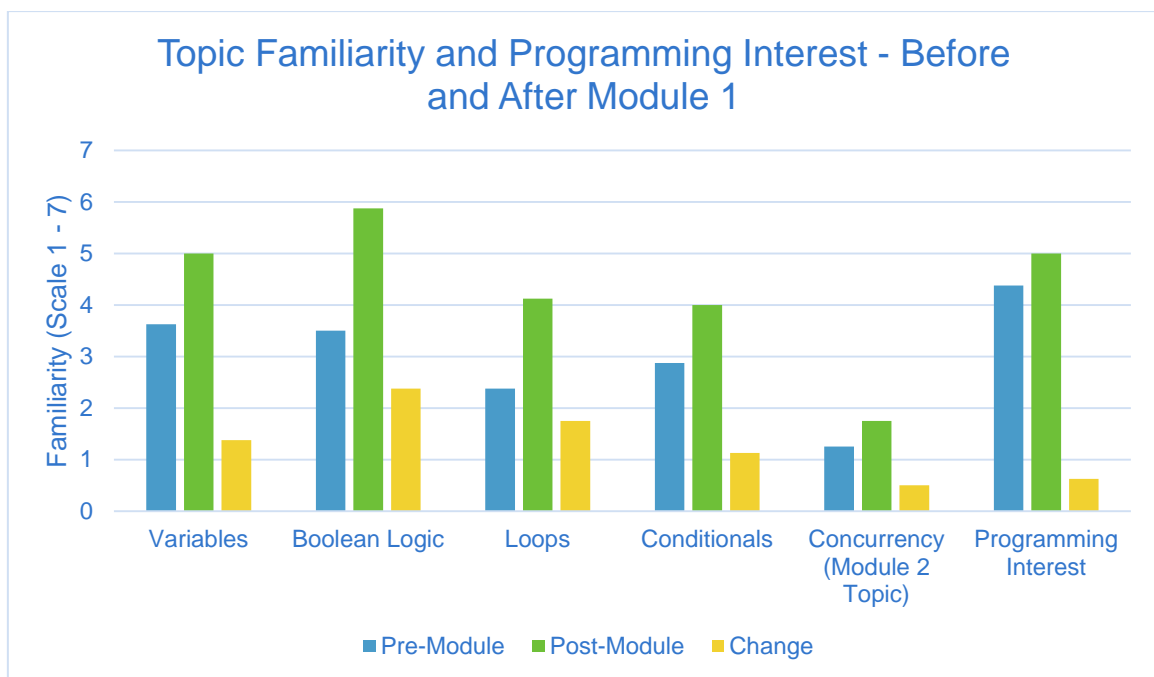


Figure 11: Change in familiarity of computer science topics and programming interest

The survey also allowed us to adjust the curriculum so that we could target it to the appropriate level and desires of the students. Most of the suggested changes came in areas where there could have been more clarity in an explanation or an addition of a picture that would help show how to organize the code most effectively. With the students' suggestions we were able to get the modules into their final state where they can be used with any

programme similar to Aiming High that wishes to expand into teaching and stimulating interest in computer science.

Discussion

In order to attempt to improve the operations of the centre this study was focused on improving computer education in the programmes that Banksia provides to the community. The Aiming High programme was identified through the interviews with Banksia's staff as an area where technology could be integrated to examine a possible extension to the educational value that the programme provides. By examining and observing the current computer science programmes we were able to identify the areas of potential improvement. These observations along with our interviews of the staff at BGCS lead us to the conclusion that a computer science curriculum based on the programming language Scratch would be the most viable option for integration into Banksia's programmes.

The computer science curriculum currently has the capacity to educate students in the Aiming High programme as well as participants of any of Banksia's computer programmes; this could benefit over 100 students throughout the year according to information from Banksia's staff. The curriculum is also made available on Banksia's website along with all of the needed assets, which means that any visitor to the site has the opportunity to complete the modules. In the future these modules may also be used by other organisations as a starting point in a complete computer science education programme. For example, the technology guides at Hume Learning Centre are in charge of running programmes for the local community with a focus on technology. These technology guides work closely with Banksia and have already shown interest in implementing our curriculum into a programme that they run with local community members. The Hume Learning Centre also works with many different community centres in addition to BGCS, where different computer education related programmes are run, in which the curriculum could potentially be integrated. The computer science modules created for study two have the potential to reach past Banksia's doors and further into the community.

Surveys of students before and after completion of the modules were completed in order to study whether the Modules helped teach students computer science concepts and whether they stimulated student's interest in the topic. Our limited results based on a small student population showed that the modules were successful in both of these areas and can be used as a tool in beginning the education process of young students who may otherwise never learn about programming. We recommend that BGCS make the curriculum available to

students via its computer club page on its main website. This will allow our modules to be accessed by anyone with an internet connection who visits the Banksia website.

Banksia's core values include the enhancement in educational opportunities for the young citizens of the area. Our creation of a first computer science curriculum allows Banksia to provide another resource for children and young teenagers of the area to develop their education. The computer curriculum provides the technological resource that Banksia needs in order to effectively equip the young people in the area with computer knowledge.

Chapter 4: General Discussion

Through two separate studies exploring the use of technology as a tool by a community organisation, we were able to propose ways that social issues could be addressed through the use of two different technology tools. BGCS was an ideal study location due to its setting in the heart of a disadvantaged community of Broadmeadows. The organisation features many programmes and we determined that GPAN and the Aiming High programme were ideal opportunities for utilising technology.

For the GPAN youth group a platform was created in order to help spread its message as part of a newly designed website in addition to hosting the Bystander Action Tool. As discussed above, the survey results of local community members showed that the webpage and Bystander Action Tool are effective uses of technology, supporting the idea that technology can indeed be used to help the missions of community organisations. This idea is further supported in the second aspect of our research with the creation of a computer science curriculum. Despite limited evaluation, it did demonstrate positive results both in introducing computer science concepts and stimulating interest in the computer science field from high school aged individuals.

Our encouraging results relate to the successes of other organisations integrating technology into their operations (Giaya, Macmillan, Price, & Roche, 2014). Previous success in using the internet to make a stand against family violence in Spain has become the benchmark of where GPAN hopes to take its new website and the Bystander Action Tool. The survey results of the computer science modules show that the new curriculum has the potential to parallel the extremely rapid growth of Scratch coding centres in the United States of America and Venezuela.

Chapter 5: Conclusion and Recommendations

As with any study, there are limitations to the scope of our studies and results. Although both studies were tested by local community members, the demographic of the participants included only residents of the surrounding community. In order to confirm our results, similar studies should be undertaken at other community organisations to increase the variation of the demographic in the study groups. Due to time constraints and number of available subjects, the sample size for the survey regarding the computer science modules is smaller than we would like. If additional time or subjects becomes available, an increased sample size could lead to results that better resemble a majority of the population. Future research investigating the use of technology at BGCS has the potential to support other programmes.

As the world of technology advances, there are ever increasing opportunities to utilize new tools to aid in the operation of community organisations. This is why we recommend continued research into how different programmes at Banksia could use tools that are provided by technology. Programmes such as the afterschool study group could use technology in the form of tablets or similar devices to decrease paper consumption of the students who complete multiple worksheets during every study group session. It is one of many examples where further research into the use of technology at the centre could bring about positive change to the operation of certain programmes held at the community centre.

Future research and realization of online advocacy programmes as well as programming based curricula should continue to help BGCS with its mission of improving the lives of the community of Broadmeadows. The two studies focused on in our research demonstrate the common underlying theme that technology can be utilized in a social environment to improve the operations of community based organisations such as Banksia Gardens Community Services and the Good People Act Now Youth Group.

References

- 2011 Census Quick Stats. (2013, March 28). Retrieved from Australian Bureau of Statistics:
http://www.censusdata.abs.gov.au/census_services/getproduct/census/2011/quickstat/2?opendocument&navpos=220
- A to Z the World. (2015). *Australia: Education Structure*. Retrieved January 22, 2015, from A to Z the World: <http://www.atoztheworld.com/>
- Australian Bureau of Statistics. (2014, October 3). *Advertised Download Speed*. Retrieved from Australian Bureau of Statistics:
<http://www.abs.gov.au/ausstats/abs@.nsf/Latestproducts/8153.0Main%20Features2June%202014?opendocument&tabname=Summary&prodno=8153.0&issue=June%202014&num=&view=>
- Australian Bureau of Statistics. (2014, February 24). *Household Internet Access*. Retrieved from Australian Bureau of Statistics:
<http://www.abs.gov.au/ausstats/abs@.nsf/Lookup/8146.0Chapter12012-13>
- Australian Bureau of Statistics. (2014, October 12). *Tullamarine - Broadmeadows (SA3)*. Retrieved from Australian Bureau of Statistics: <http://stat.abs.gov.au>
- Barriball, K. L., & While, A. (1994, February). Collecting data using a semi-structured interview: a discussion paper. *Journal of Advanced Nursing*, 19(2), 328-335.
- Barth, M., & Gridley, H. (2008). Families living in poverty in Broadmeadows: Challenges, survival strategies, and support services. *Australian Community Psychologist*, 20(1), 36-46.
- Broadmeadows, Geelong Properties Attract Interest. (2013, May 23). *Australian Business Intelligence*.
- Carlton, M. (2003). *Motivating Learning in Young Children*. Retrieved from National Association of School Psychologists:
http://www.nasponline.org/resources/home_school/earlychildmotiv_ho.aspx
- Cooper, S., Dann, W., & Pausch, R. (2003). Teaching Objects-first in Introductory Computer Science. *34th Annual Technical Symposium on Computer Science Education* (pp. 191-195). New York, NY: ACM.
- Coslovich, G. (2006, 10 21). Reinventing Broadmeadows; Urban Renewal. *The Age*, p. 5.
- Crime Statistics by LGA*. (2014). Retrieved from Victoria Police:
http://www.police.vic.gov.au/content.asp?a=internetBridgingPage&Media_ID=72178
- Crime Statistics Official Release*. (2014). Retrieved from Victoria Police:
http://www.police.vic.gov.au/content.asp?a=internetBridgingPage&Media_ID=72176

- Croom, K., Lewis, D., Marchell, T., Lesser, M. L., Reyna, V. F., Kubicki-Bedford, L., . . .
 Staiano-Coico, L. (2010, August 6). Impact of an Online Alcohol Education Course on Behavior and Harm for Incoming First-Year College Students: Short-Term Evaluation of a Randomized Trial. *Journal of American College Health*, 57(4), 445-454.
- Data By Region*. (2014, June 23). Retrieved from Australian Bureau of Statistics:
<http://stat.abs.gov.au/itt/r.jsp?databyregion>
- Department of Employment. (2014). *Small Area Labour Markets Australia*. Commonwealth of Australia.
- Domestic Violence Statistics*. (2015). Retrieved from Domestic Violence Prevention Centre:
<http://www.domesticviolence.com.au/pages/domestic-violence-statistics.php>
- Education and Work, Australia*. (2014, December 12). Retrieved from Australian Bureau of Statistics: <http://www.abs.gov.au/ausstats/abs@.nsf/mf/6227.0>
- Giaya, D., Macmillan, C. T., Price, A. S., & Roche, M. S. (2014). Creating Engaging Computer Programs for Children in an After-School Computer Club. (*Undergraduate Interactive Qualifying Project No. E-project-121414-190803*). Retrieved from Worcester Polytechnic Institute Electronic Projects Collection:
<http://www.wpi.edu/Pubs/E-roject/Available/E-project-121414-190803/>
- Good People Act Now (GPAN)*. (2013). Retrieved from Banksia Gardens Community Services: <http://banksiagardens.org.au/community-support-broadmeadows/good-people-act-now-gpan/>
- Is Coding the New Second Language?* (2014, January 27). Retrieved from Coder in Africa:
<http://www.coderina.org/is-coding-the-new-second-language/>
- It Gets Better Project*. (2014). (Savage Love LLC.) Retrieved from It Gets Better:
<http://www.itgetsbetter.org/>
- Kafai, Y. B., Peppler, K. A., & Chiu, G. M. (2007). High Tech Programmers in Low-Income Communities: Creating a Computer Culture in a Community Technology Center. *Communities and Technologies* (pp. 545-563). London: Springer-Verlag London Limited.
- Key Statistics*. (2014). Retrieved from Domestic Violence Victoria:
<http://www.dvvic.org.au/index.php/understanding-family-violence/key-statistics.html>
- King, E. (2015). *State Budget Submission 2015-16*. Retrieved from Victorian Council of Social Services: <http://vcoss.org.au/state-budget-submission-2015->

16/?utm_source=website&utm_medium=featured-top&utm_campaign=state-budget-submission-2015-16

- Langdon, D., McKittrick, G., Beede, D., Khan, B., & Doms, M. (2011). *STEM: Good Jobs Now and for the Future. ESA Issue Brief #03-11*. Washington, DC: US Department of Commerce.
- Lawson, R., & de Matos, C. (2000). Information technology skills in the workplace: Implications for Bachelor of Arts degrees. *Australian Journal of Educational Technology, 16*(2), 87-103. Retrieved from <http://www.ascilite.org.au/ajet/ajet16/lawson.html>
- Lifelong Kindergarten Group. (n.d.). *About Scratch*. Retrieved February 14, 2015, from Scratch - Imagine, Program, Share: <https://scratch.mit.edu/>
- Lombardo, C., Zakus, D., & Skinner, H. (2002). *Youth Social Action: Building a Global Latticework Through Information and Communication Technologies*. Great Britain: Oxford University Press.
- Magrane, C. (2015, March 30). *Scratch Being Taught in Caracas Venezuela*. Retrieved from ScratchED: <http://scratched.gse.harvard.edu/stories/scratch-being-taught-caracas-venezuela>
- Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning program with scratch. *Proceedings of the 39th SIGCSE technical symposium on Computer science* (pp. 367-371). New York, NY: ACM.
- Meloni, J. (2010). Technologies for Teaching Online. *The Chronicle of Higher Education*(57.11). Retrieved January 22, 2015
- Puente, S. N. (2011, June). Feminist Cyberactivism: Violence Against Women, Internet Polotics, and Spanish Feminist Praxis Online. *Journal of Media & Cultural Studies, 333-346*.
- Ricon, J. d.-O. (2015, March 13). Banksia Gardens Community Services: Overview. (K. Naugler, Interviewer)
- Schwartz, B. (2013). *Four Ways Technology is Being Used to Solve Problems around the World*. Retrieved from Today Made: <http://todaymade.com/blog/four-ways-technology-is-being-used-to-solve-problems-around-the-world/>
- Shaw, M. (2006, January 12). Melbourne Biscuits Going Chinese; Kraft Move Takes 150 Jobs. *The Age*, p. 2.

- Song, R., Spradlin, T. E., & Plucker, J. A. (2009). The Advantages and Disadvantages of Multiage Classrooms in the Era of NCLB Accountability. *Center for Evaluation & Education Policy*, 7(1), 1-6.
- Stone, S. J. (1994). Strategies of teaching children in multiage classrooms. *Childhood Education*(71.2), 102+. Retrieved January 24, 2015
- Support Banksia Gardens Community Services. (n.d.). Retrieved from Banksia Gardens Community Services: <http://banksiagardens.org.au/support-banksia-gardens-community-services/>
- The Joint Task Force on Computing Curricula. (2001). *Computing Curricula 2001*. ACM, IEEE.
- The Joint Task Force on Computing Curricula. (2013). *Computer Science Curricula 2013*. ACM, IEEE.
- UN Women. (2014, October). Retrieved from Facts and Figures: Ending Violence against Women: <http://www.unwomen.org/en/what-we-do/ending-violence-against-women/facts-and-figures>
- Victims Support Agency. (2012). *Victorian Family Violence Database Volume 5: Eleven-year Trend Analysis: 1999-2010*. Melbourne, VIC: Department of Justice.
- Victoria Economy One of the Worst. (2014, November 22). *The Age*, p. 6.

Appendices

Appendix A

Website Template

GPAN



Appendix B

Questions for Focus Group with GPAN Members

1. What is the intended audience for this site?
2. What themes and colours for the webpage would correspond best with existing GPAN materials?
3. What content do you feel should be included in the website, such as images, videos, and links to other resources?
4. What content do you feel is most important, and should be included on the homepage?
5. How often will each section of the webpage need to be updated?
6. How many people will need administrative access to the page for updates?
7. How much web development or web administration experience do these individuals have?

Appendix C

GPAN Website Usability Study: Pre-Survey Commentary and Survey Questions

Pre-Survey Commentary:

Your answers to this survey are optional and if at any point you do not want to answer a question or no longer wish to participate, you have no obligation to do so. Your responses will remain completely confidential. The results of this survey will be used to help create an advocacy webpage that will be tailored to the responses of the survey questions and needs of the community.

Survey Questions:

General Website

1. Rate the ease of use of the website as a whole (Scale 1-7)
2. What were any locations that were confusing or unintuitive? (Open ended)
3. What improvements could be made to improve usability? (Open ended)

Homepage

1. Rate the ease of use of the homepage (Scale 1-7)
2. From the home page, how clear is the purpose of each other page in the menu? (Scale 1-7)

Get Smart

1. Rate the ease of use of the Get Smart page (Scale 1-7)
2. How informative is the page about GPAN's mission and goals? (Scale 1-7)

Get Help

1. Rate the ease of use of the Get Help page (Scale 1-7)

Get Acting

1. Rate the ease of use of the Get Acting page (Scale 1-7)

Get Along

1. Rate the ease of use of the Get Along page (Scale 1-7)

Bystander Action Tool

1. Rate the ease of use of the Bystander Action Tool (Scale 1-7)
2. How informative did you feel the videos were? (Scale 1-7)
3. After seeing the videos, how likely are you to act differently if encountering any of the scenarios depicted? (Scale 1-7)

Appendix D

Pre and Post Video Consent Forms

Videotape Consent Form

As part of our work for Banksia, we would like to video tape you. By agreeing to be recorded on video, you acknowledge that your identity may be publically disclosed and the information you provide in the video or audio may be used on the Banksia website, in a published report, and/or presentations on this work.

I have read this form and have the opportunity to ask questions about the project.
I agree to these terms.

Printed Name

Signature: _____

Date: _____

Post Video Release Form

Thank you for participating in the video. We have benefited greatly by creating this video and hope to impact others using it. We would now like to reflect on the video and determine if you would allow us to use the video you participated in. By signing this form you agree to allow this video to be used in a research project. You also understand that the content may or may not be used on a public website.

I agree to the terms.

Printed Name

Signature: _____

Date: _____

Appendix E

Computer Science Curriculum Pre/Post-Module Assessment

1. What is your age?
2. What is your gender?
3. What is your date of birth? (Used for creation of a unique id code to collate results)
4. What is your favourite colour? (Used for creation of a unique id code to collate results)
5. Rate your familiarity with the following computer science concepts:
Scale: 1 (Completely Unfamiliar) – 7 (Very Familiar)
 - a. Variables
 - b. Boolean Logic
 - c. Loops
 - d. Conditionals

The following questions are to be used only in the Post-Module Assessment

6. Did you complete the module? (yes/no)
7. How enjoyable did you find the module? (Scale 1 - 7)
8. How difficult did you find the module? (Scale 1 - 7)
9. How boring did you find the module? (Scale 1 – 7)
10. How interested are you in computer science? (Scale 1 – 7)
11. What could be changed to make it more entertaining? (open ended)

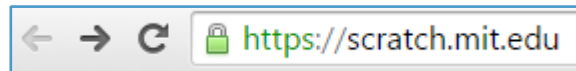
Appendix F

Getting Started with Scratch

Getting Started with Scratch

1. Opening Scratch

- a. Open your web browser
- b. Go to <https://scratch.mit.edu>

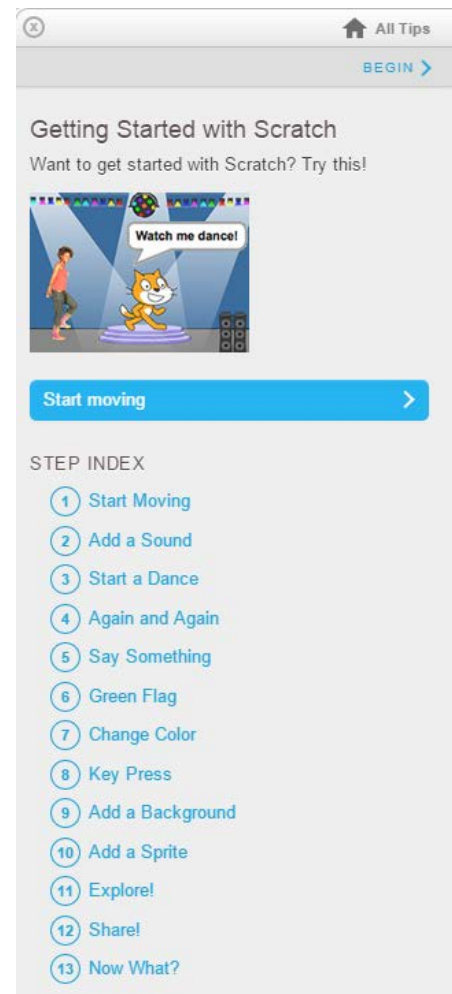


- a. Click “Create” at the top of the webpage



2. Beginning the “Getting Started with Scratch Tutorial”

- a. Follow the getting started guide at the right-hand side of your screen to create your own cartoon!



Appendix G

Computer Science Module 1

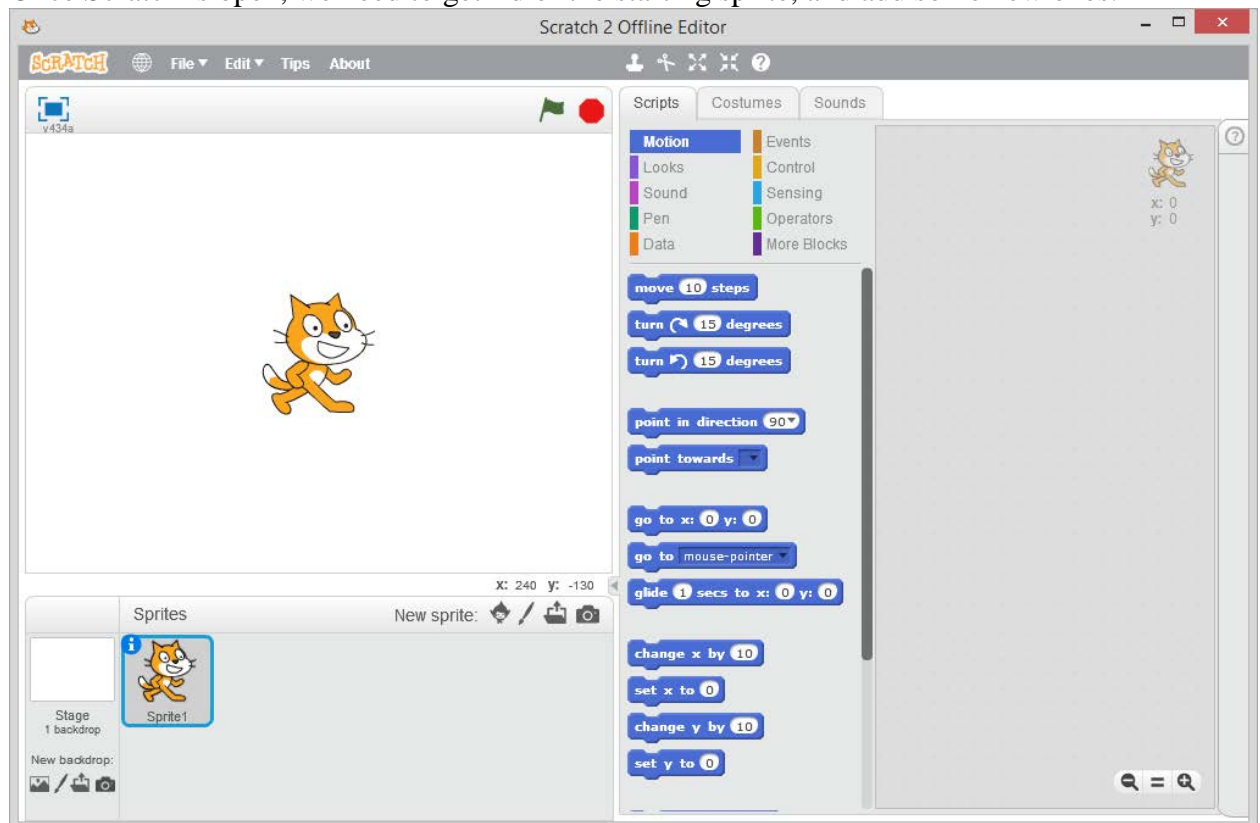
Module 1

Pong

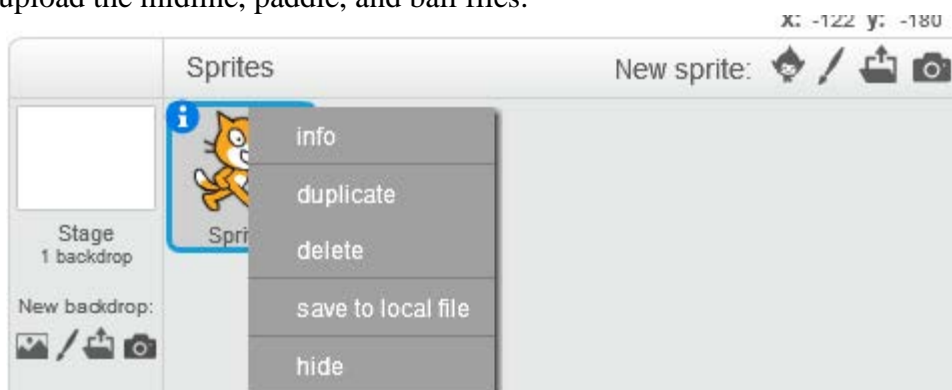
This module, we'll learn how to create Pong in Scratch.

1. Setting the Playing Field

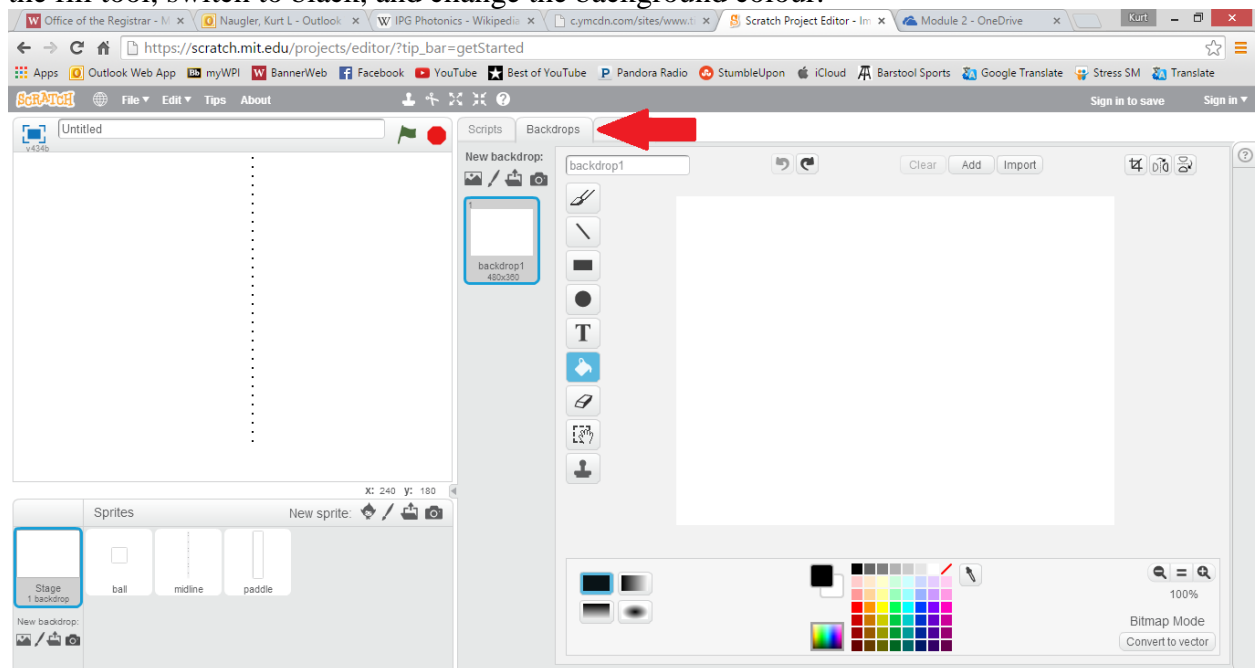
Once Scratch is open, we need to get rid of the starting sprite, and add some new ones.



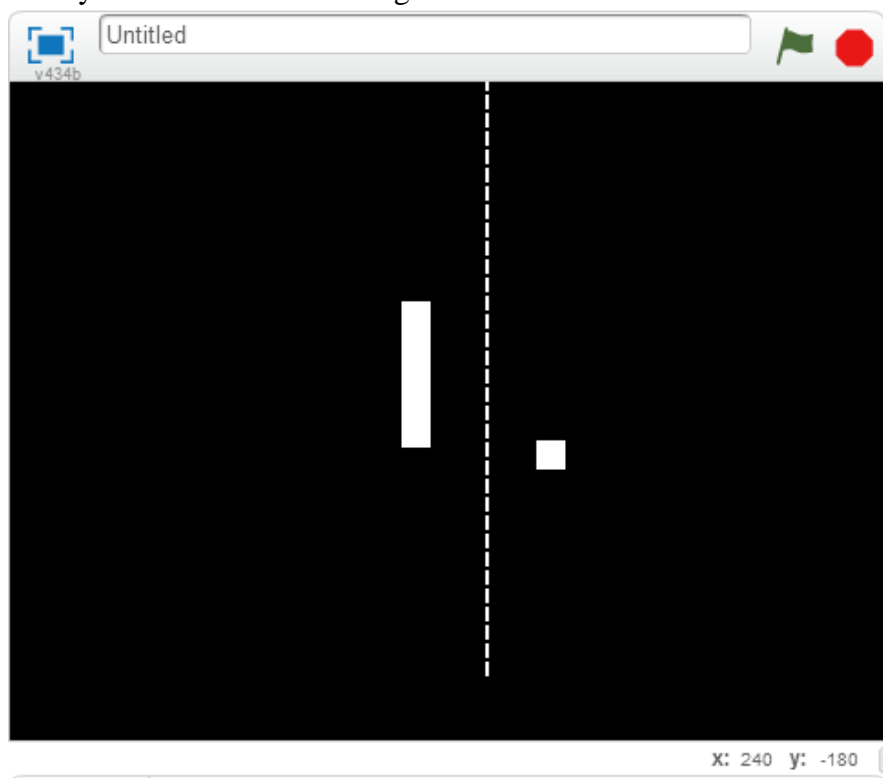
Right click the existing sprite and select delete. Then to add the new sprite, click to the folder icon in the “Sprites” section of the screen and navigate to the Assets folder. Then upload the midline, paddle, and ball files.



But what's this? You only see a few dots on the screen? That's because Pong has a black background. Go to the “Backdrops” tab and select the fill tool (the bucket of paint). Select the fill tool, switch to black, and change the background colour.



Now you should see something like this:



We still need to add another paddle. Right click the paddle sprite and select duplicate. Now you've got all the pieces you need for Pong!

But there's still one problem. Nothing is in the right spot! We could drag them into position by hand, but there's a better way to do this. Before you move on though, save your project!

2. Assigning Positions


We're going to put everything in its place using some of Scratch's motion commands. Start by going back to the scripts tab, and then select midline in the Sprites window. Now every script you make will apply to the midline. First we're going to add the

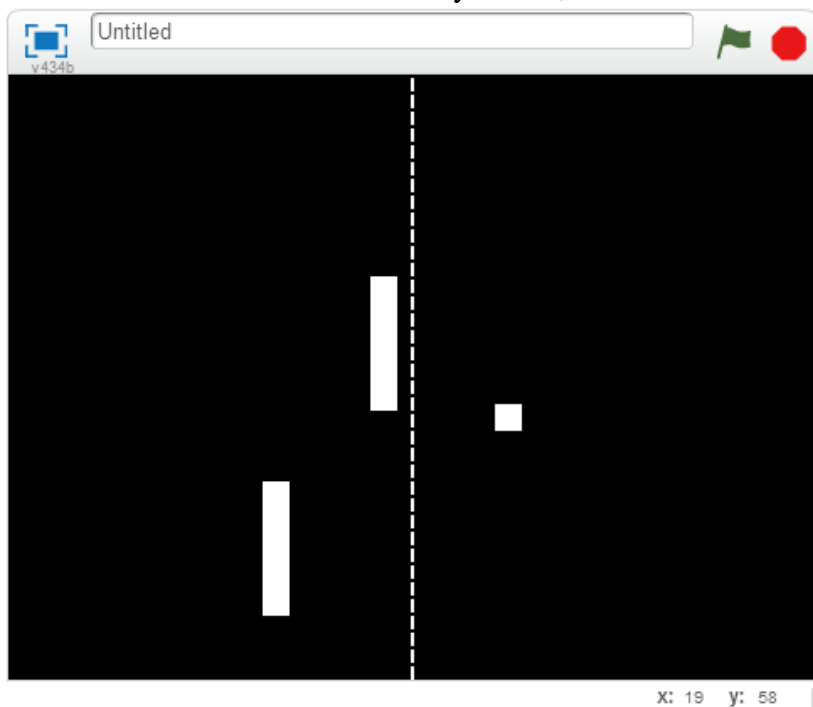
command to set everything up at the start. Go to the events tab and drag the “When pressed” command into the scripts window.

This **event** makes any code attached to it execute when the start flag at the top of the screen is pressed.

Now we need the code that actually moves the midline. Go to the motion tab, and drag the “go to x: ? y: ?” code right below the starting event. It should look like this:




Make sure to enter 0 as both the x and y values, and then hit  to see it in action!



Now we need to do the same for the ball and paddles. First, select the ball in the sprites panel, and then use the same commands and position values as for the midline. Next, we need to add a property for the ball so that when we add movement later, it'll bounce.



Now, if you run the script again, you'll see the ball move to the middle of the screen.

For the paddles, it'll be a little different. Select one, and then click the  in the corner of the sprite.




Now rename the paddle to *Left Paddle*. This will make sure we don't get the two mixed up, because the code for each will use different numbers.

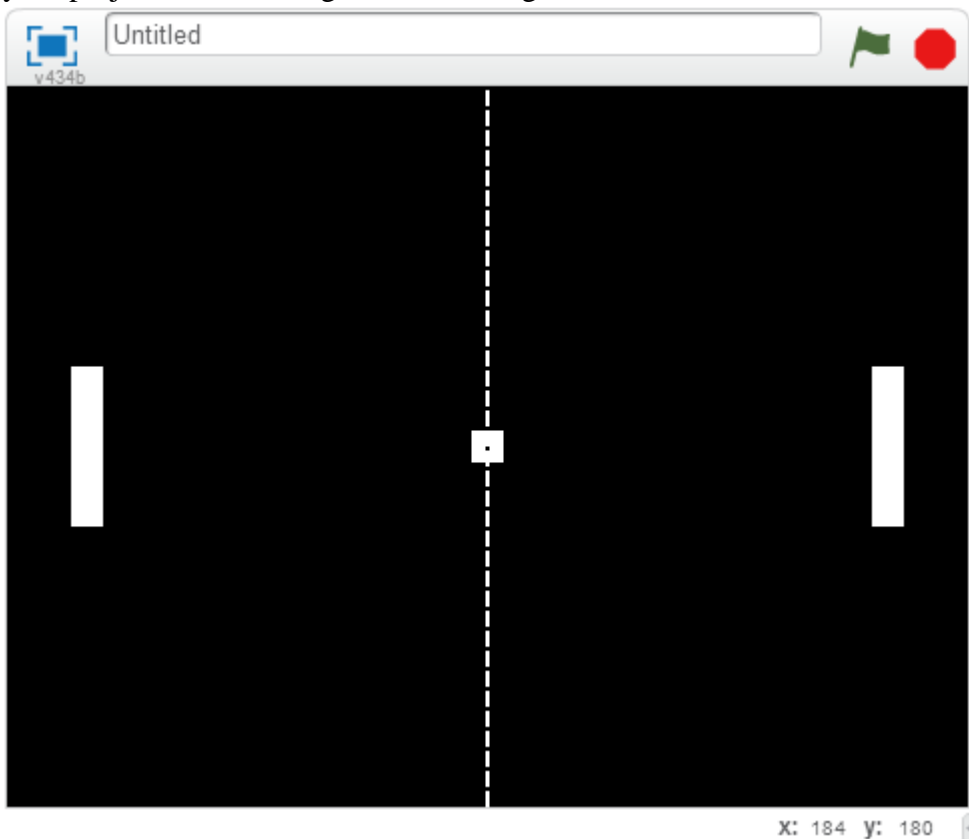
Now we can go and put in the same code we used for the ball and midline, except here we'll use a different pair of coordinates. The x position will be -200, and the y will be 0.



Now select the other paddle, and repeat the same process. Rename it to *Right Paddle*, and then add the code to move it into position. It will be at an x position of 200 and a y position of 0.



Once you press , all the pieces are in their starting points! Now after you save your project, it's time to get them moving!



3. Moving the Paddles

First, select the left paddle, and then go to the control tab and drag over the “forever”



block.

Info

- This block is a variant of a **while loop**, one type of **conditional**. A **while loop** will execute the code inside of it over and over until whatever condition is controlling it is false. The forever block's condition is just set to true, so it keeps going forever, or until the script stops.
- Other scratch while loops are the wait, repeat, repeat until, and wait until blocks.


This will make sure the script is always checking to see if we're trying to move the paddle. Now we're going to put two “if then” blocks one after another inside the forever block. These two blocks will check whether the W and S keys are pressed, to control the paddle moving up and down.



Info


- The “if then” block is an **if statement**, the other type of **conditional**. An if statement will execute the code it contains if and only if its condition is true.

Now we need to add sensing! Go to the Sensing tab in scratch and drag the “key space

pressed”  [NOTE: if right size put in place of “key space pressed”] into the conditions for the “if then” blocks. Then set the top one to “w” and the bottom to “s”. These will be the up and down commands for the player on the left. Finally, we need to actually make the paddle move. Add the block to the inside of both “if then” blocks.

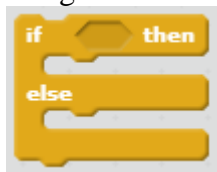
You should have something like this:



Now if you press  it, you can move the left paddle up and down by pressing w and s! If you play around with it, you might run into a problem like this:



You can't just have paddles running off the screen! To fix this we'll add some checks using an "if then else" statement.





Info


• **If then else statements** are the bigger brother of **if statements**. Instead of doing nothing if their condition is false, they will run the code in the **else** section of the statement.

Place **if then else** blocks into both if then blocks, so we can check both when moving up and down. Then move the change commands into the **if then** section of the blocks.




Next we'll add the conditions for each of the blocks. For the top block, we need to check that the position of the paddle is in relation to the top of the screen, which is at $y = 180$.

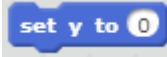
To do this we'll use the  command from the operators tab. This will check if the number on the left is **less than** the number on the right, and if it is it will return true. Drag this into the condition slot. On the left side we want to use the  block from the Motion tab, and on the right, the value we compare it with.

To figure out what this needs to be, we need to do a little math. The top of the screen is at 180, but the  gives us the middle of the paddle. We don't want it to go off screen, so we need to take into account the full size of it. We also need to consider how far the paddle can move each time the loop is executed, and we only want to let it move if it can go the full distance.

$$y\ position + \frac{paddle\ height}{2} + paddle\ speed < 180$$

When we plug in all the numbers, we get that the y position must be less than 130 for the paddle to move. Put this number into .



Now we need to tell the program what to do when the condition isn't true. We want the block to move like it should toward the edge, but not go off-screen. So we'll use the  block, and set the value in it to 140. Place this into the **else** section. We should have something like this:


```

if y position < 130 then
  change y by 10
else
  set y to 140

```

Now we need to do the same thing when moving down. The only difference here will be using the `>` command from the operators tab, and using negative numbers instead of positive. This will check if the number on the left is **greater than** the number on the right, and if it is it will return true.

We're finished script for the left paddle, and we should now have something like this:

```

when clicked
  go to x: -200 y: 0
  forever
    if key w pressed? then
      if y position < 130 then
        change y by 10
      else
        set y to 140
    if key s pressed? then
      if y position > -130 then
        change y by -10
      else
        set y to -140

```

There's one last thing we need to do for the paddle, which is to store its speed, so we can use it when the ball bounces of it. Go to the Data tab and click the "Make a Variable" button. Name it *Left Speed* and make sure "For all sprites" is selected. This means we can use it in scripts for any sprite.

Info

- A **variable** is used to store some sort of data. It could be a number, a set of letters, or even both. We can then access that data by name and change or use it.

```

set Left Speed to 0

```

We'll now put the `set Left Speed to 0` command into each of the **if then else** statements, right after we change y by either 10 or -10. We'll set the value of the first

set Left Speed to 0

to 10, and the second to -10. We'll add two more into the **else** sections, setting the speed to 0. The script should now look like this:



We can reuse this code for the right panel. Click and drag the forever block over the right paddle icon in the Sprites panel, and then drop it there. This copies it over to the other sprite. Switch to that sprite's view, and then connect the forever block to the bottom of the script.

If you run the script now, you'll notice that both paddles will move up and down when you use "w" and "s". We need to change the keys the script is looking for from "w" and "s" to "up arrow" and "down arrow". We also need to create a new variable,

set Left Speed to 0

called *Right Speed* and change the commands to act on *Right Speed*. The script should look like this:



Once you save the game, all that will be left is to write the script for the ball!

4. Getting the Ball Rolling

The ball needs to do a few things in Pong. It needs to start in a random direction, and then bounce off the paddles and top and bottom. And if it goes past either paddle, the other player needs to get a point (or in our case, win).

The first thing we want to add after clicking on the ball is a short delay, so the game doesn't start immediately after we hit . Go to the control tab and select the



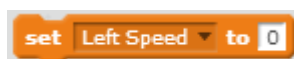
command, and then attach it to the bottom of the script and set it to 3.

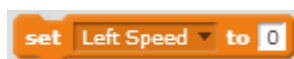
Now we want to set and store the balls initial speed. We'll create two variables, this time choosing "For this sprite only". Name them *XSpeed* and *YSpeed*.




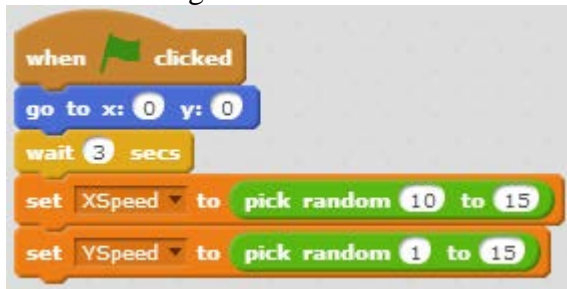
Variable name:




For all sprites For this sprite only





Now, we need to add  twice after the wait command. We set the first to be for *XSpeed* and the second to be for *YSpeed*. We'll then go to the operators

tab and grab . This selects a random number between or including the two in the command. Drop one of these into each of the set commands. For the first one, set the values to 10 and 15. For YSpeed, set them to 1-15. It should look something like this:

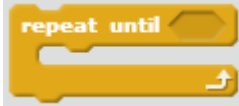



Now, we need to make negative values possible. We'll use a pair of **if then** statements for this. Add these to the script now. Add the  comparator to each of the **if** statements, and set the right side of each to 1. Then, put  into the left side of both 's, and set them to be from 0 to 1.

In the body of the first if then, we need to add another , set to change *XSpeed*. We then need to drag a multiplication operator,  into the value slot. We will set the right side to -1. For the left, we will drag the *XSpeed* block from the Data tab. This will set *XSpeed* to negative its value if the random number picked is 1. We then do the same with the second **if then** statement, replacing *XSpeed* with *YSpeed*.



Now we'll add the code to make the ball move at that speed. We'll use the repeat until

block from the Control tab.  For the condition, we need to check if the ball has passed either the left or right paddles. Add the  operator from the Operators tab to the condition.

Info

- OR is a **Boolean operator**, meaning it evaluates values of **true** and **false**. OR evaluates as **true** when one or both of the values it is operating on is **true**. Otherwise it evaluates as **false**.

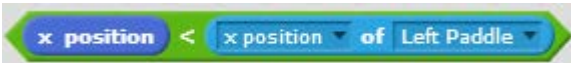
- Ex:

- true OR false is true
- false OR false is false

On the left side we will check the ball's position against the position of the left paddle. Add a **less than** operator, and make the left side the **x position** value from the Motion tab. The right side will use a new block from the sensing tab that can be used to access the values held by each sprite.



Set the drop down boxes to "x position" and "Left Paddle". The comparator should look like this:



For the right side of the **OR** statement, add a greater than comparator, and then duplicate the **x position** and **x position of Left Paddle** blocks from the left side. The greater than statement should look like this:



Now we can make the ball move. Add a change x by and a change y by blocks to the inside of the loop. Set the values to *XSpeed* and *YSpeed* respectively.

This will make the ball move. If you run the game now, you'll notice that the ball will move and then slide into a corner. This is because we haven't implemented bouncing yet. To get our ball to bounce, we need to add three **if then** statements, one for each paddle, and one for the edge. For each if statement, add a touching block



, and set the drop down values to "Left Paddle", "Right Paddle", and "edge" respectively. Now, the inside of the loop will look like this:

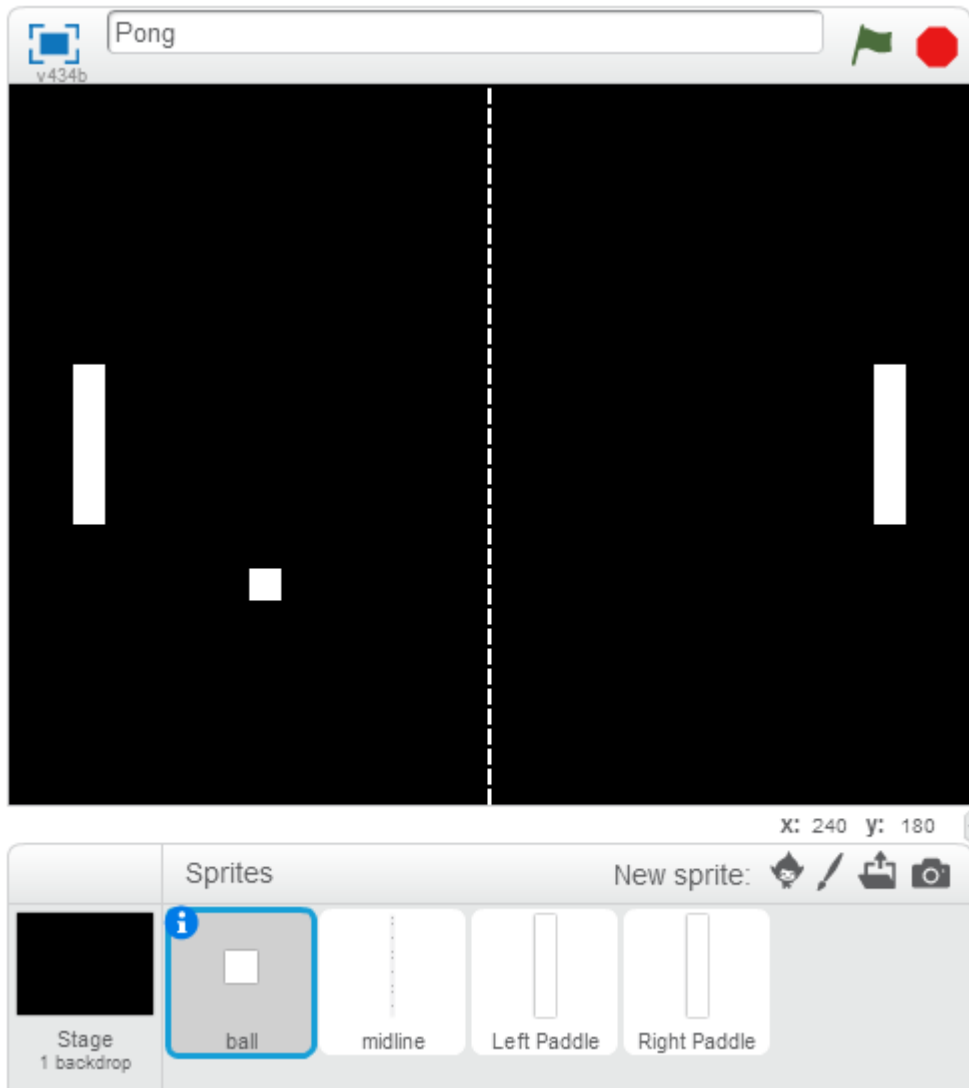


In the "Left Paddle" **if then** statement, add a set command set to *XSpeed*, and with a value of $XSpeed * -1$. Then add another set command that sets *YSpeed* to $YSpeed + Left Speed$, the variable we defined with the Left Paddle. Do the same thing for the

“Right Paddle”, replacing “Left Speed” with “Right Speed”. These will make the ball bounce off the paddle, and if the paddle is moving, add a little speed in that direction. The third **if then** controls bouncing off of the top and bottom of the screen. Here we want to set *YSpeed* to *YSpeed* * -1. Now the inside of the loop should look like this:



CONGRATULATIONS! You have just programmed your own version of Pong! Your final version of the game should look like this:



In this module, you have learned to use programming concepts such as: Variables, Conditional Statements, Loops, and User Interaction.

Bonus work: For an additional challenge try using the additional files in the assets folder to add a scoreboard to your game by using the number images as new sprites. You can also change the code to use the space key to start the game. Lastly, you can add sprites that will appear at the end of the game, showing when either player 1 or player 2 wins.

Appendix H

Computer Science Module 2

Module 2


Space Invaders

Timers

The first thing we have to get setup is the background. Change the colour of the background to black, just like when making Pong. Also, similarly to when making Pong, you will also need to delete the existing sprite. Once you've done that, change back to the "Scripts" tab.

Setting up the Variables

This will be where we run any scripts that control the game as a whole. The first script we build will set up all the **global variables**. **Global variables** are variables that are available to

all sprites. Add a "When  clicked" event, and then create all the variables listed below and set them to the corresponding starting values by adding a "set VARIABLE to" block to the script.

Variable	Starting Values	What it does
score	0	Keeps track of your score
timer	0	Keeps track of how many times out timer script has run
laser_offset	10	How far from the aliens we create their laser bolts
lives	3	How many lives the player has left
alien_direction	1	What direction the aliens are moving in (-1 is left, 0 is down, and 1 is right)
alien_speed	0.5	How fast the aliens move
laser_0_countdown	Random from 15 to 120	How long until the aliens shoot their first type of laser
laser_1_countdown	Random from 15 to 120	How long until the aliens shoot their second type of laser.

Once you've created and set those variables, the script should look like this:



But we're not done yet. We'll also need to use a different type of data, **lists**. These will store some information about the different rows of aliens we're going to have.

Info


- **Lists** are kind of like variables, but they can store more than one thing. These can be numbers, or strings of letters. You access each item using an **index**, or the number the item is in the list. If you want to get the first item, you'd use "item 1 of list", and if you wanted the third you'd use "item 2 of list"
- In other programming languages, lists might be called arrays, and the first thing stored is located at 0 instead of 1.



We'll need to create seven different lists to keep track of different information about the aliens. Go to the "Data" tab and click "Make a list" to create the following lists.

List	What it does
alive_1	Keeps track of whether row 1's aliens are still alive
alive_2	Keeps track of whether row 2's aliens are still alive
alive_3	Keeps track of whether row 3's aliens are still alive
alive_4	Keeps track of whether row 4's aliens are still alive
alive_5	Keeps track of whether row 5's aliens are still alive
x_pos	Keeps track of the X position of each

	column of aliens
fire_pos	Keeps track of the Y position we create lasers at for each column

But lists are a little harder to initialize than variables – we might still have entries left over from running the program before. But there’s an easy way to clear them using the “until” loop.

We add a loop for each list, and put a  block in each. Make sure to change the list selected so we have one for each. Then for the condition, we add an

 block, set the right side to zero, and the left to , also found in the “Data” tab. Your script should look like this now:




This will go through, set the variables to their starting values, and then for each list, delete the first value and then loop, and keep deleting until the list's length is zero, meaning it's empty. This works because when you delete an entry in the list, every entry after it moves forward to fill the space.

But we're still missing one part – a way for all the sprites to know that the startup is complete! We'll use **messages** for this.

Info

- **Messages** are ways for different scripts to communicate, and keep in sync. If we don't keep scripts in sync, they could try to change the same variable at the same time, or they might just start too early or too late.
- Keeping the different scripts in sync is known as **concurrency**, and it is becoming more and more important as computers get more advanced and have more processor cores.

A Scratch 'broadcast message' block, which is orange with a white border. It contains the text 'broadcast' in white and 'message1' in black, followed by a small downward-pointing triangle.

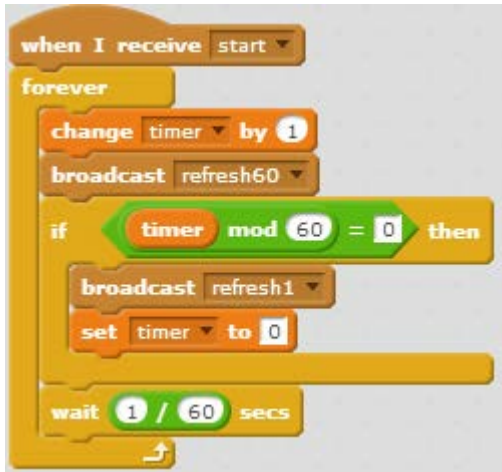
Add the  block at the end of the script, and create a new message called *startup*. The completed script should look like this:



Keeping Everything Together

The next script we write here will be a **timer** script. This will be our solution for **concurrency**. The script will send messages at two different rates – 60 times per second, and 1 time per second. The first is fast enough to give us smooth movement, because at that point each movement blurs together into the next. Another option could be to use 30 messages per second instead, but movement can still look jumpy. The 1 time per second message will be used for slower changes, such as switching the costumes on the sprites.

The completed timer script looks like this:



The event starting this will be a new message you create, *start*. This message will be sent when you hit the spacebar to start the game. Then we have a forever loop, which will increase the timer variable each time through, and broadcast the *refresh60* message. The script checks if *timer mod 60* is equal to 0, which is the shorthand way of asking whether there is a remainder from *timer* divided by 60. If the remainder is zero, then we broadcast the *refresh1* message, and reset *timer* to 0. We then wait $1/60^{\text{th}}$ of a second. The *refresh60* message will be our 60 times per second update, and the *refresh1* message will be our one time per second update.

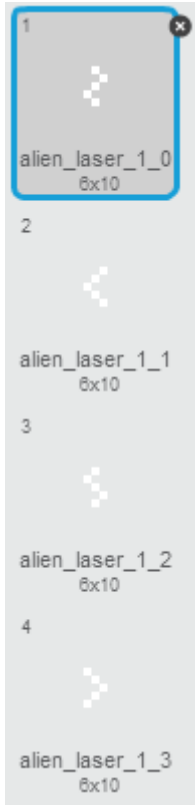
Making the Aliens Shoot

You might ask why we're making the aliens shoot before we've even added the alien's yet. The answer is because we are controlling when they shoot lasers from the background. First off, we need to upload the sprites for the two different lasers by using the upload sprite from file button. Upload the *alien_laser_0_0.png* and *alien_laser_1_0.png* files. Then rename the first to *alien_laser_0* and go to its costume tab. We now need to upload its other costume, *alien_laser_0_1.png*. The costumes should now look like this:



Now we need to do the same thing for *alien_laser_1_0*. Rename it to *alien_laser_1* and upload its costumes: *alien_laser_1_1.png*, *alien_laser_1_2.png*, and *alien_laser_1_3.png*.

Then reorder them so they look like this;



Now we need to actually use them. This will use a process called **cloning**, where the script makes a copy of a sprite all by itself. The script that does this is below and should be created in the script section for the background:



Every time it gets the *refresh60* message, the script will check if either *laser_0_countdown* or *laser_1_countdown* are equal to 0. If they are, it will create a clone of the matching laser, and reset the countdown to a random number between 15 and 120. Then both countdowns will be decreased by 1. This makes each of the lasers shoot anywhere from every quarter of a second to every two seconds.

Wrapping up the Stage

Now we only have two very short scripts left for the background. These'll wrap everything up when you either win or lose. You will need to create 2 new messages, *win* and *game_over*. When you get the *win* message, the script will wait one second, and then stop all scripts:



When you get the *game_over* message, the script will do the same thing:



Now, make sure to look over all the scripts we've covered so far and make sure you have them, and then save!

Starting the Game

If you run the program now, you'll notice that you can't see anything happening. This is because we still don't have anything sending the start message, which we need to kick the timer and everything else off. This is where our next sprite comes in. Upload the *start.png* file, and go to its script tab. Make sure to also go to its info page and uncheck the "show" option!

We're only going to have one script here:



When it gets the startup message, the sprite will go to the center of the screen, show itself, and then wait for you to press space. When you do, it'll disappear and send the start message. Run the program now to make sure it does this, and then save!

Building the Tank

Now we're going to add the player's tank to the game. Upload the *tank.png* sprite. We're also going to need the laser so we can make clones of it now, so upload *tank_laser.png* too. We're going to need to have responses for a few messages for the tank – *startup*, *start*, *win*, *game_over*, and *refresh60*. We'll start off with the short ones.

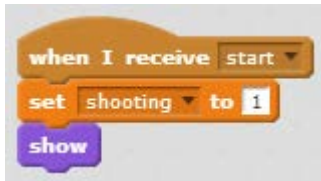
Startup

The startup is extremely simple. We just need to hide the tank during the start screen, and move it to the correct position:



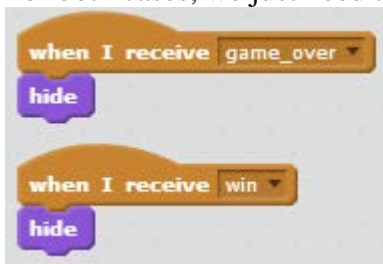
Start

When the game starts, we need to show the tank. We're also going to make a new variable for this sprite called *shooting*, and set it to one. This variable makes sure that you need to press the spacebar once for each shot, instead of being able to hold it down to shoot. The script should look like this:



Win and Game Over

For both cases, we just need to hide the tank:



Refresh60

This is where things get complicated. We're going to need to check several different cases for the tank – if the left or right arrows are pressed for movement, if the spacebar is pressed for firing, if the spacebar *isn't* pressed to reset the shooting variable, and whether the tank has been hit by one of the alien's lasers. First off, add the message's event trigger:



Now, we'll add the if statements for movement:

```

if key left arrow pressed? then
  if x position > -218 then
    change x by -5
  else
    set x to -223
endif
if key right arrow pressed? then
  if x position < 218 then
    change x by 5
  else
    set x to 223
endif

```

The first if statement just handles moving left, and makes sure you won't go off of the edge of the screen. The second does the same thing for moving right. If you run the code now, you should be able to move left and right.

Now we'll add the code for shooting:

```

if key space pressed? and shooting = 0 then
  set shooting to 1
  create clone of tank_laser
endif
if not key space pressed? and shooting = 1 then
  set shooting to 0
endif

```

The first if statement makes sure that you can only shoot if you press the spacebar and are not currently shooting, so you can't just hold down the spacebar to keep shooting – you have to press it for each shot. To do this we'll set shooting to 1 when we fire, and then create a clone of the tank laser. Actual movement of the laser will be handled by the laser sprite itself.

The second statement resets the variable to zero if you have released space and shooting was set. If you run this code now, you should see more clones of the laser appearing around the screen unless you've hidden the laser, but they won't move just yet.

The final component will check if the tank has been hit by either of the alien's lasers:

```

if touching alien_laser_0 ? or touching alien_laser_1 ? then
  change lives by -1
  if lives < 0 then
    broadcast game_over
  else
    broadcast die
endif

```

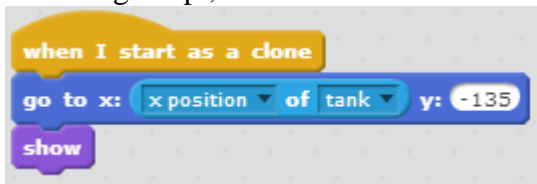
This decreases the amount of lives left, and then ends the game if all your lives are gone. Otherwise, it just broadcasts the *die* message, which will let the life indicator know to update. Now that you've finished up the tank's code, we can move on to its laser!

Shooting for the Stars


With the tank's laser, we're going to use a new block:



This block runs the attached script when you create a clone of another script. It'll be how we initialize each laser bolt, as well as how we create all the aliens later. Use it to make the following script, that moves the new laser to the tank's position and makes it show up:



Now we need to add the events for *game_over* and *win*. Here, we'll just delete both clones

using , which you can find in the "Control" tab:





The only message left to address is *refresh60*. The laser needs to move and check if it's touching any enemies or the edge. We also need to check if the laser destroyed the last alien when it is touching an edge, so we can send the *win* message. The script to do this looks like this:



First, this script checks if it is touching either red, white or the edge. To get the white colour, click on the coloured square in the "touching colour" block, and then click on any of the white sprites. To get the red, we need to upload the *mothership.png* file as a new sprite. Then, set the block to the mothership's colour.

If this is true, we know we've hit something, so the laser can disappear. But before that happens, we want to check if there are any aliens still left alive. If there aren't, we can end the

game. We use a couple of logical operators to do this, the  and  operators. By combining the or operators with the "contains" block, the combined block will be true if any of the alive lists contains a 1, indicating that at least one alien is alive. We then

use the not operator to negate the value, so it will only be true if none of the lists contain 1, which only occurs if all the aliens are dead, in which case we send the *win* message. When the laser isn't touching red, white, or the edge, we just move it up by 10 pixels. If you start up the game now, you'll be able to shoot with the space bar. Make sure to save before moving on.

INVASION

Now that we have a laser, we need some enemies to aim at! We're going to add the top row of aliens. Upload *invader_2_0.png* as a new sprite, and rename it to *row_5*. Next, duplicate this sprite four times, renaming them *row_4*, *row_3*, *row_2*, and *row_1*. We're not going to use these though, except as placeholders so we can use their names in *row_5*'s scripts. Later, we'll delete and replace them with the actual rows 1 through 4. Then add *invader_2_1.png* as a second costume. The costumes menu should like this:



But right now we only have one alien. We need a whole row of them. We'll first get through the *startup*, using the following script:



When the program starts up, we'll hide the alien, and then move it to the right height on the screen. This'll make sure that when we check the height of the sprite and its clones, we'll get the right number. Then we just set a few variables, created just for this sprite. First is *clone_counter*. This is how we count how many clones we've made, and how we identify each clone. Next is *costume*, which keeps track of which costume the sprite is using, so we can tell which one to switch to. And finally we set *boom*, that we use to keep track of if the alien has been hit, and change the **state** of the script while the explosion plays.

Now that you've finished creating this, we'll need to use it again for other sprites. Click the Backpack at the bottom of the screen, and drag and drop the above block of code into it. This will save it, so you can add it elsewhere.

Next we're going to add the script to make the first clone. We'll need this again for the rest of the aliens, so add it to the backpack too:

```

when I receive start
  create clone of myself

```

We only create one clone here, because we're going to use **recursion** to create the rest.

Info

- **Recursion** is where one script or block of code causes itself to happen. In this case, we're going to create another clone in the script that runs when a new clone is created. This lets us reuse the same event code multiple times, and is an alternative to using a loop.
- When using **recursion**, make sure to have some way for the script to stop! Otherwise the code will keep running forever, and crash when it uses up too many resources!

We're going to use the following script to do this:

```

when I start as a clone
  go to x: -175 + clone_counter * 35 y: 95
  change clone_counter by 1
  insert x position at clone_counter of x_pos
  insert 1 at clone_counter of alive_5
  show
  if clone_counter < 11 then
    create clone of myself

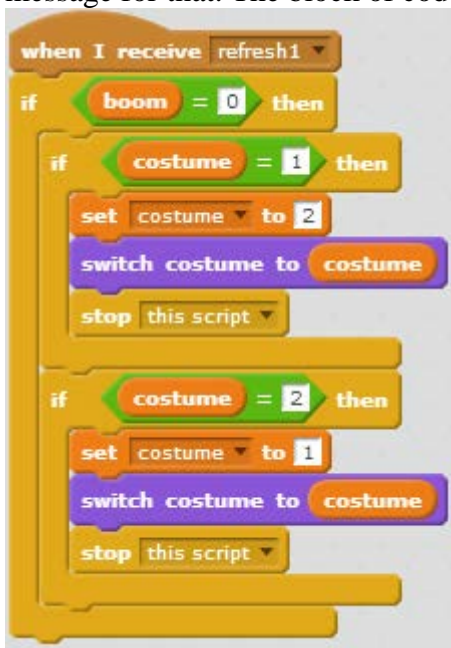
```

This starts off by moving the new clone to its starting position. We use *clone_counter*, which starts at 0, to determine this. The first clone will just start at an x-position of -175, but because we increment the counter, the next will be at -140. After we increment the counter, we add the alien to the list of x-positions, and to the alive aliens in row 5. We're using the counter as an **index** to the list, and because we're not going to change it again for this clone, we can keep using it to get the same position in the list. This takes advantage of the fact that a clone copies all of its local variables from the one copied, but any changes that occur afterward to the original don't affect the clone. This means that the original alien will have a counter value of 0, the first clone a value of 1, the second 2, and so on. The last clone will end with a counter value of 11, ending the recursion because *clone_counter* < 11 will no longer be true. This code will be almost the same for each row of aliens, so add it to the backpack.

If you run the code now, you'll see that the top row of aliens will appear!



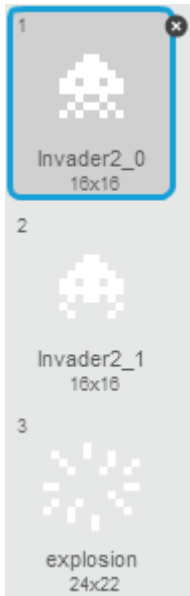
The next step is to make their costumes cycle. We'll finally make use of the *refresh1* message for that. The block of code is pretty simple:



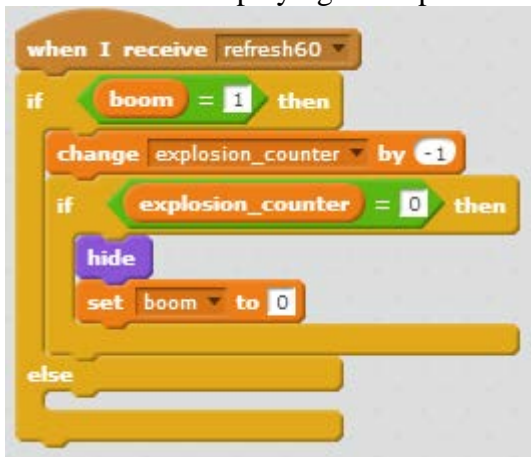
First off, we check if the alien is currently exploding. If it is, we don't want to change the costume, because that will change it away from the explosion visual. Then, we need check whether which costume it is, and then switch it to the other. We then stop the script, because if we don't, when the costume is 1, it would switch to two, then continue on and check if it is costume 2, and immediately switch it back to costume 1. This would just make the sprite flicker each second, and that just doesn't look good.

Run your code now, and your aliens should be switching back and forth between the costumes. And make sure to add it to your backpack! We can reuse this again.

Earlier I mentioned the explosion visual. We should upload this now as another costume, because we'll be added the explosion visual, along with everything else for the *refresh60* message. The file to upload is *explosion.png*. The sprite's costumes should now look like this:

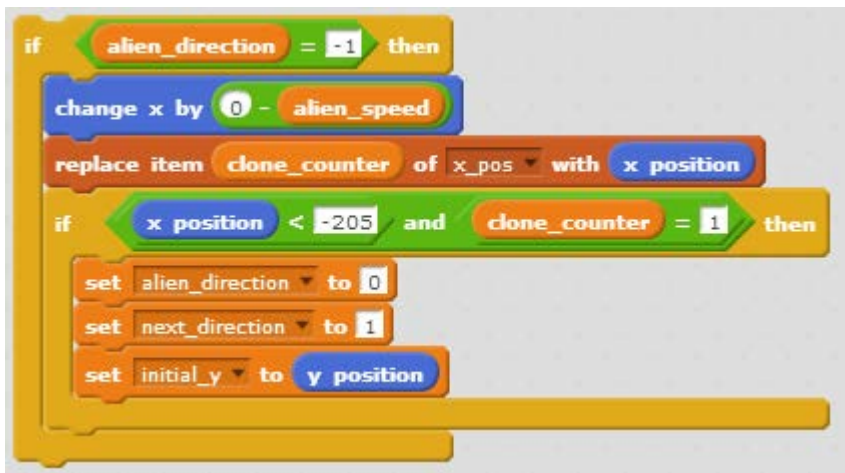


Now, we're going to start building the code. First, we'll set up the message event and the code to handle displaying the explosion:



Here we create a new variable only for this sprite, *explosion_counter*. It keeps track of how many refreshes until the alien is finished exploding, and we can hide it. We decrease it by one, and then check if the counter is down to zero. If it is, we hide it and then set boom back to zero. This is important, because we use the top row of aliens, specifically the top left, to control the movement of the rest of the aliens. We can't just delete the clone if it's destroyed, because we then end up with the rest of the aliens not knowing when to change direction. And we can't leave *boom* set to one, because then the movement code in the else section of the conditional won't run. So we just hide the alien and reset the variable to zero. Add this to the backpack – this part will be the same for all of the aliens.

Next we'll start making the movement code. We'll break this up into a set of if statements that we'll put one after another in the else section. First off we have the code that moves the alien left.



This will change the x position of the alien by the opposite of the alien speed, and update the position in the list. If this is the top left clone, we'll also check if it has reached the left boundary, at $x = -205$. If it has, we set the direction to zero (indicating down), and set *next_direction* (a new variable for this sprite) to 1, indicating right. We'll use this to set *alien_direction* when it's finished moving down. We also set another new variable for this sprite, *initial_y*, to the current position of the alien. This will tell us how far we are from the original height as we move down, so we know when to stop.

The next block gets kind of crazy. As we move down, we need to



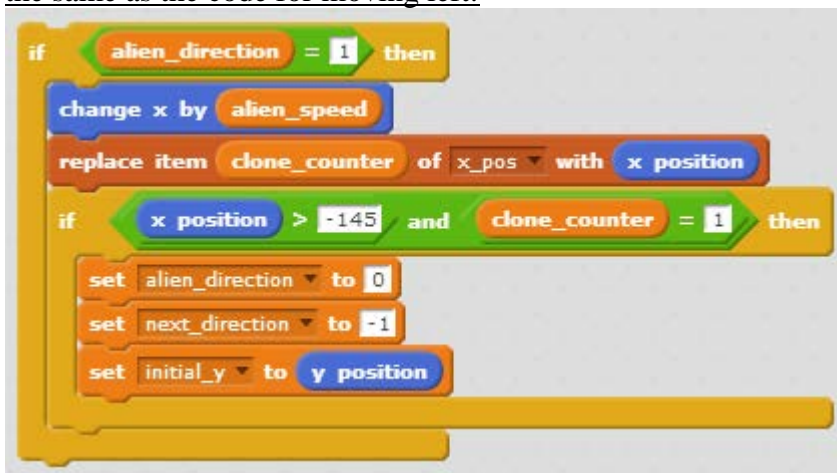
check how far we've gone down, and changed the values in the *fire_pos* list as we do. But we can't just change all the values, because then we'll be changing the values we've set to show that all aliens in the column are dead, and we shouldn't shoot lasers from there anymore. So we need to check if this alien's spot in any of the alive lists are set to one.

We first move the alien down by the alien's speed. Then we check if any of the aliens in this column (the column number is stored by *clone_counter*!), and if there are, we update the entry in *fire_pos* so we know the new location to create the alien's lasers.

Then, if this clone is the top left one, we check if the alien's moved down 10 pixels from where it started. If it did, then we set the next direction to move, and increase the speed a little bit.

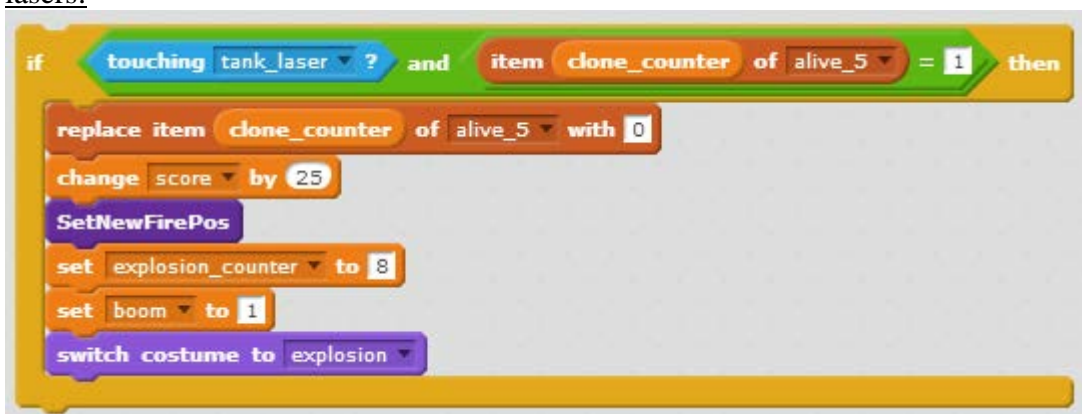
Then, we check if the row has reached the tank's level, to see if the player has lost. If there are any aliens left alive in this row, and they've gotten within 25 pixels of the tank's height, we'll send the *game over* message.

In the next block of code with the else, we control the robot moving right. This is pretty much the same as the code for moving left:



We change the position, update the x position, and then, if this is the first clone, we check if we're at the rightmost edge of movement. Then we set the direction to down (0) and the next direction to left (-1), and store the current y position.

The last block in the else statement handles what to do if the alien is hit by one of the tank's lasers:



Here we check if the alien is touching the laser and is currently alive, so we won't get it exploding again if it already exploded. Then we set the alien to dead and increase the players score.

To make the next block, we're going to need to go back to the background view. This is because we need to write some code that works for all five rows, but unfortunately, if we write it while in one of the rows, we won't be able to select that row in the script. So, we'll write it outside of any row and drag and drop it into all of the rows we make. The next block

is a custom block. Go to the "More Blocks" tab, and then click . Then type in *SetNewFirePos* as the name, and then click "OK". This'll create a block that looks like this:



This is where you add code that you can reuse by adding the new block that's in the menu on the right. We'll add the content to it now, which will set a new fire position in the column when the alien is destroyed.

```

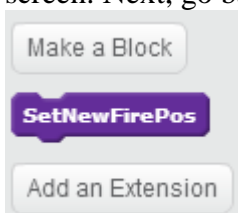
define SetNewFirePos
if item clone_counter of alive_1 = 1 then
  replace item clone_counter of fire_pos with y position of row_1 - laser_offset
else
  if item clone_counter of alive_2 = 1 then
    replace item clone_counter of fire_pos with y position of row_2 - laser_offset
  else
    if item clone_counter of alive_3 = 1 then
      replace item clone_counter of fire_pos with y position of row_3 - laser_offset
    else
      if item clone_counter of alive_4 = 1 then
        replace item clone_counter of fire_pos with y position of row_4 - laser_offset
      else
        if item clone_counter of alive_5 = 1 then
          replace item clone_counter of fire_pos with y position of row_5 - laser_offset
        else
          replace item clone_counter of fire_pos with 999

```

The code checks the alien at *clone_counter* in each row, and if the alien is dead, moves on to the next. If the alien is still alive, it sets *fire_pos* to the y position of that row minus an offset. If all the aliens in the row are dead, then it sets the fire position to 999.

Now that you've made this we don't want to change anything in it. If you do when it is in one of the row's, you might find that you can't select your current row in the drop downs. Drag and drop it into the backpack, and then delete it from the background. We don't need it there anymore.

Now, go back to *row_5*, and drag the block we just stored in the backpack back to the scripts screen. Next, go back to the "More Blocks" tab, and get the block we generated there:



Add it to the script we had been working on, so it looks like this:



Then, the code sets the explosion countdown, and sets boom, so that the counter begins to count down. The costume is then set to explosion.

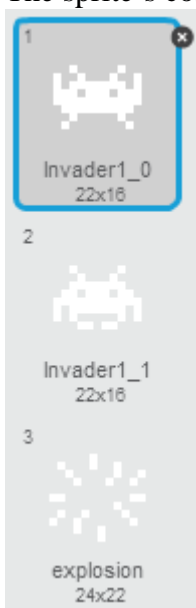
If you run the code now, the top row of aliens will move back and forth across the screen, making their way to the bottom. And if you shoot them, they'll explode! Now we just need to add a little bit of code to handle the game over scenario:



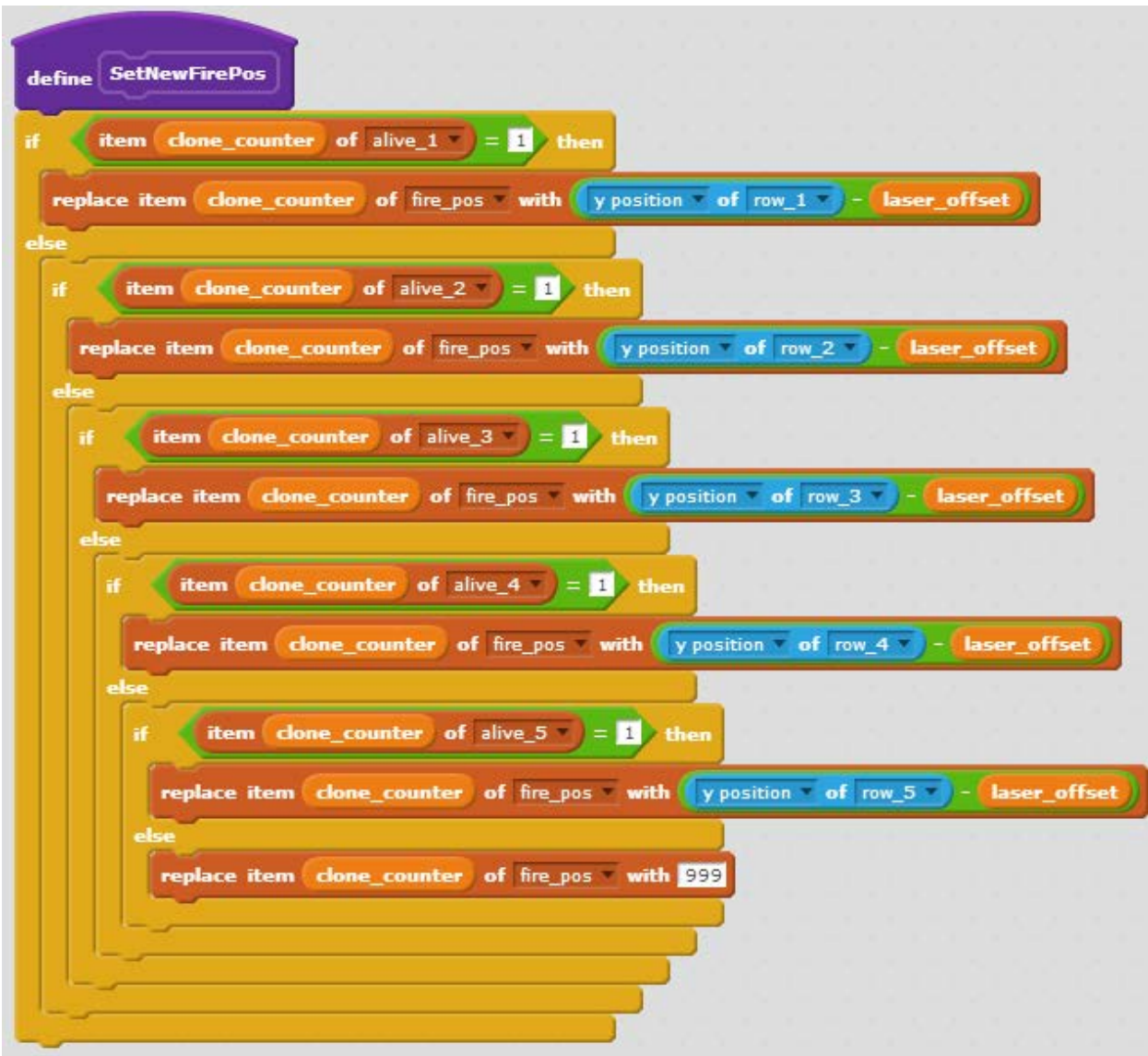
We don't actually need to worry about the win scenario, because all the aliens have to have been destroyed.

Adding the other Aliens

Now we're going to add the middle rows of aliens. We'll start off with the second from the top, row 4. Delete the previous *row_4*, and upload the *invader_1_0.png* file, and rename the created sprite to *row_4*. Then, upload *invader_1_1.png* and *explosion.png* as new costumes. The sprite's costume list should look like this now:



Now we're going to add in the blocks stored in the backpack that don't need to be changed. These are the SetNewFirePos definition:



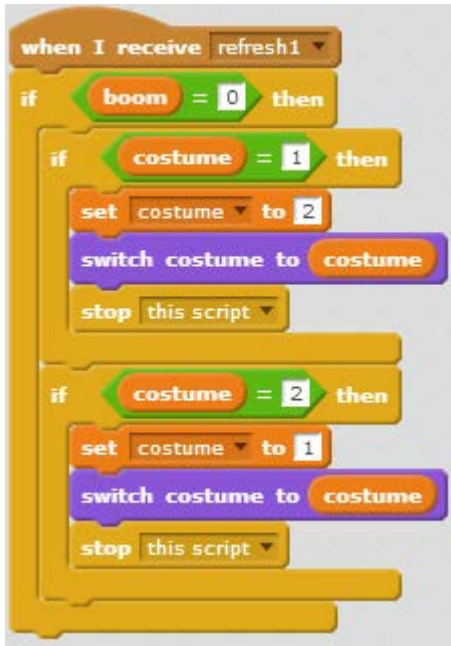
```
define SetNewFirePos
  if item clone_counter of alive_1 = 1 then
    replace item clone_counter of fire_pos with y position of row_1 - laser_offset
  else
    if item clone_counter of alive_2 = 1 then
      replace item clone_counter of fire_pos with y position of row_2 - laser_offset
    else
      if item clone_counter of alive_3 = 1 then
        replace item clone_counter of fire_pos with y position of row_3 - laser_offset
      else
        if item clone_counter of alive_4 = 1 then
          replace item clone_counter of fire_pos with y position of row_4 - laser_offset
        else
          if item clone_counter of alive_5 = 1 then
            replace item clone_counter of fire_pos with y position of row_5 - laser_offset
          else
            replace item clone_counter of fire_pos with 999
```

The *start*, and *game_over*, and *refresh1* blocks:



```
when I receive start
  create clone of myself

when I receive game_over
  hide
  delete this clone
```



The next blocks we saved, we need to make some slight modifications to. First, there's the startup code. You just need to make a little number change – the one you saved sets the y position to 95, we need to change it to 70, so it looks like this:

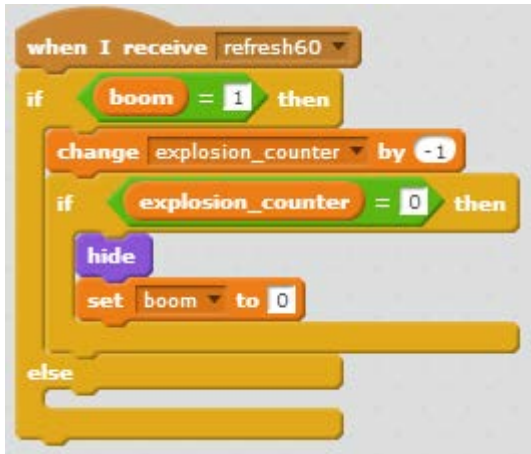


When a new clone starts up, we need to change things a little more. Drag it in, and then change the y value in the go to block from 95 to 70. Then, change the list in the “insert 1 at clone_counter” block from *alive_5* to *alive_4*:



Now we're almost done with this sprite, and we haven't had to do much at all! We just need to add in the saved *refresh60* block, and fill in the else part! Drag in the block, so it looks like

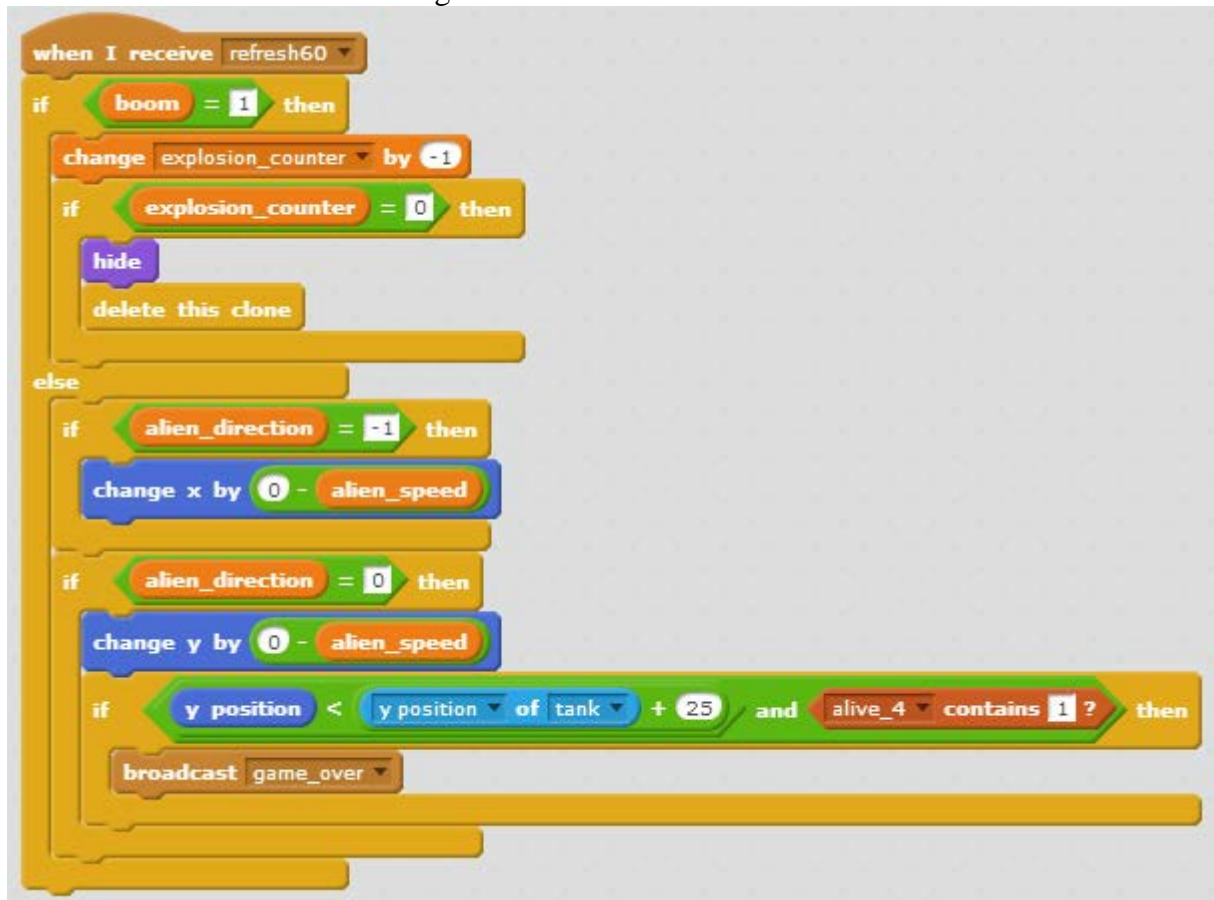
this:



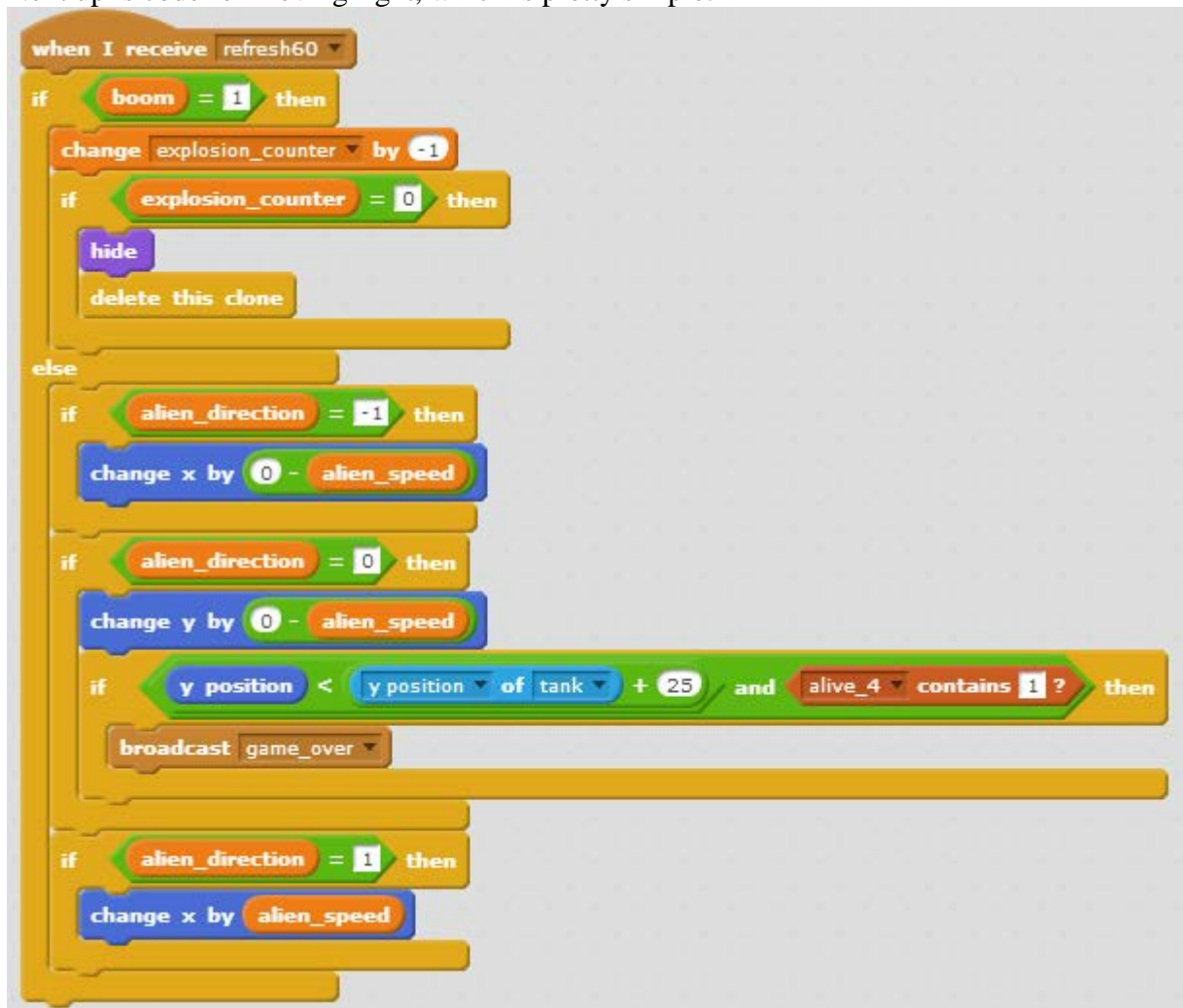
Now, we'll add the code to move left, which is a cut down version of the code for row_5. The *refresh60* script should look like this now:



Now, we'll add code to move down. This will just move the alien down, and then check if the row has reached the tank and the game is over. The block should look like this now:

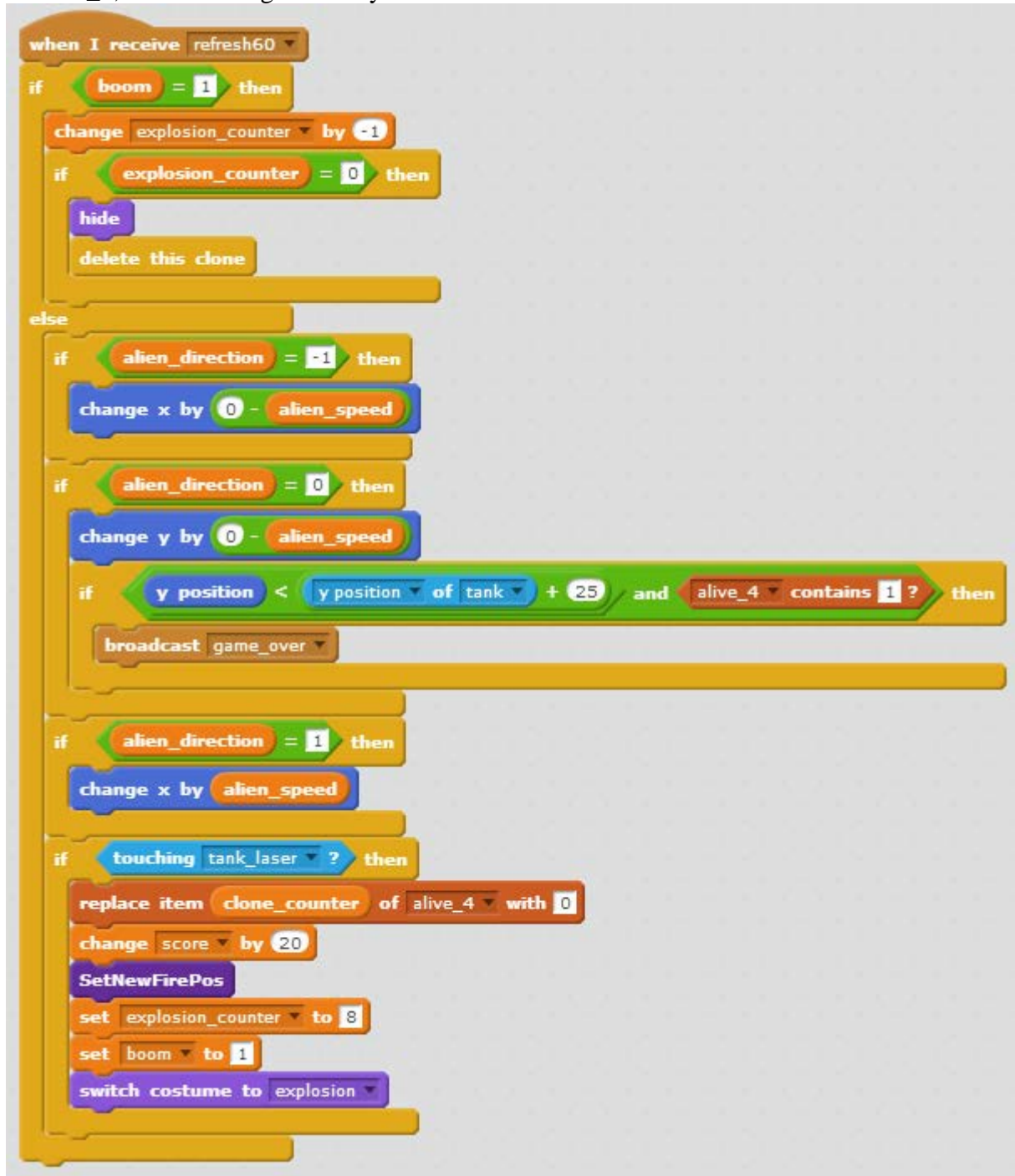


Next up is code for moving right, which is pretty simple:



We just change x by the alien speed. Now we'll add the code for the alien being hit by a laser, which is exactly like the code we used for the top row, except we use the list *alive_4* instead

of *alive_5*, and we change score by 20 instead of 25:



Now that we've finished this one, we can move on to the next row. This time, we're just going to duplicate the sprite. Delete the placeholder *row_3*, and then right click *row_4* and select duplicate, and rename the new sprite to *row_3*. Now, all we need to change are a few numbers and lists.

First, in “startup”, we’re going to change the y position from 70 to 45, so it looks like this:



Next, in “When I start as a clone”, we’re going to change the same thing, as well as switch the list from *alive_4* to *alive_3*:



Finally, in the *refresh60* script, we need to change all the uses of *alive_4* to *alive_3*. We also need to change the score increase from 20 to 15.

```

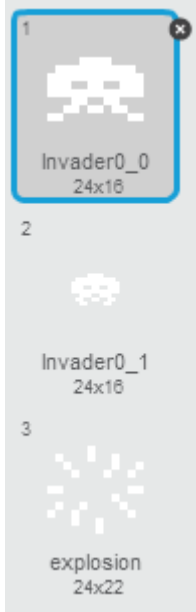
when I receive refresh60
  if boom = 1 then
    change explosion_counter by -1
    if explosion_counter = 0 then
      hide
      delete this clone
    else
      if alien_direction = -1 then
        change x by 0 - alien_speed
      if alien_direction = 0 then
        change y by 0 - alien_speed
      if y position < y position of tank + 25 and alive_3 contains 1 ? then
        broadcast game_over
      if alien_direction = 1 then
        change x by alien_speed
      if touching tank_laser ? then
        replace item clone_counter of alive_3 with 0
        change score by 15
        SetNewFirePos
        set explosion_counter to 8
        set boom to 1
        switch costume to explosion
  
```

Look at that, we just added an entire other row of aliens, and we didn't even have to write any more code!

Making the last two rows will be a little more difficult, because they use different sprites. Delete the placeholder *row_2*. Duplicate *row_3*, and rename it *row_2*. Now, go to the costumes tab and delete both of the alien sprites. Make sure to leave the explosion sprite! The costume list should look like this now:



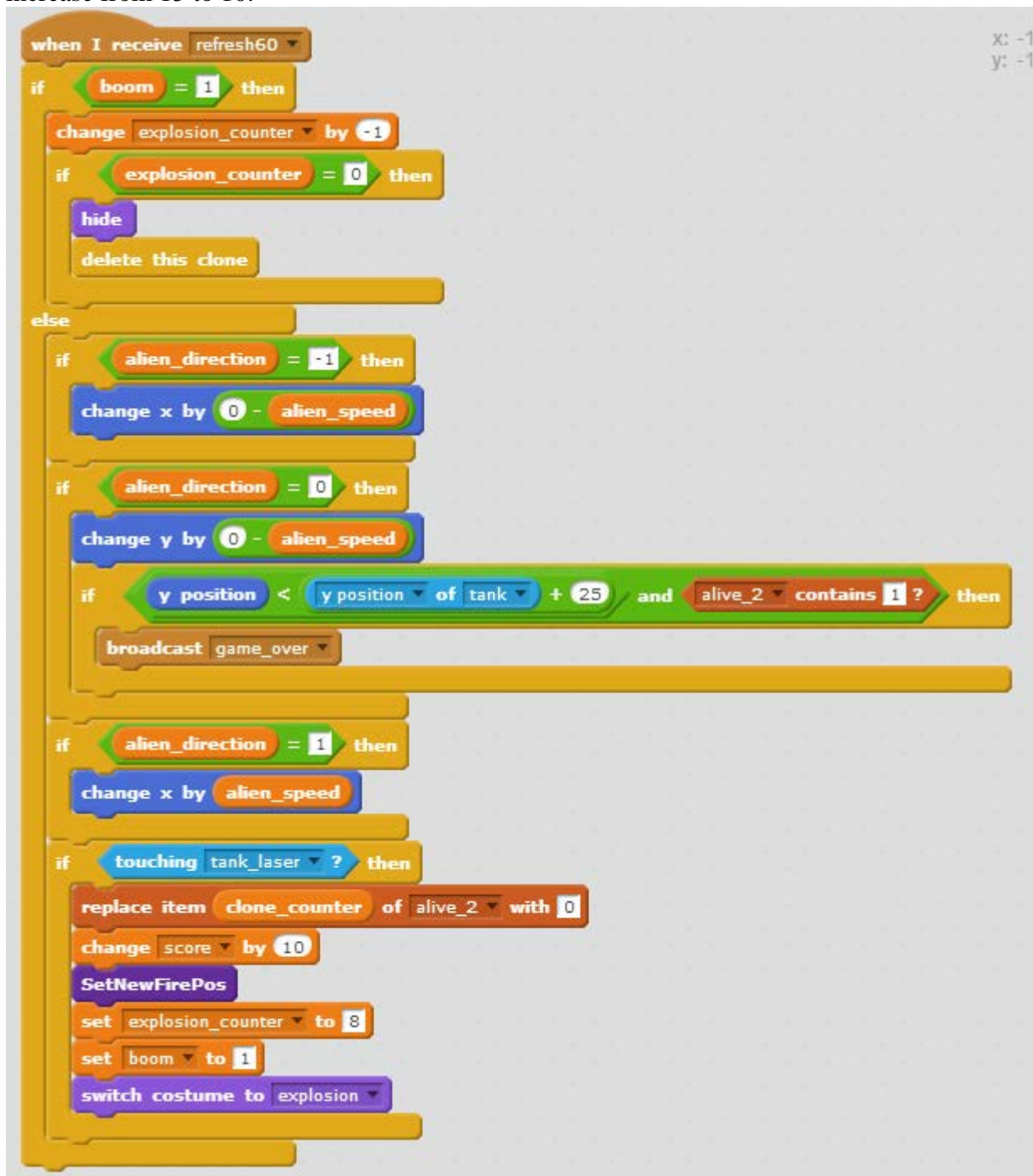
Now, we'll upload *invader_0_0.png* and *invader_0_1.png* as new costumes and arrange them so they look like this:



Now we just need to update the same scripts as last time. First, we'll change the y position in "Startup" and "when I start as a clone" from 45 to 20, and update the list to *alive_2*.



Then, we'll update the lists in "refresh60" from *alive_3* to *alive_2*, and change the score increase from 15 to 10.



Another row finished quickly! To finish off, delete the placeholder *row_1*, and then duplicate *row_2*. Rename the duplicate row to *row_1*. Here, we'll make the same change to startup –

change the y position from 20 to -5:



```
when I receive startup
hide
set y to -5
set clone_counter to 0
set costume to 1
set boom to 0
```

Next, we'll update "when I start as a clone". We'll need to change the y position in the first line to -5, and the lists to *alive_1*. We also need to add one block right before we insert 1 into *alive_1*.



```
insert y position - laser_offset at clone_counter of fire_pos
```

This block sets the initial values of *fire_pos* to row 1, so that until any aliens are destroyed, it will look as if the row 1 aliens are shooting. The updated code should look like this:



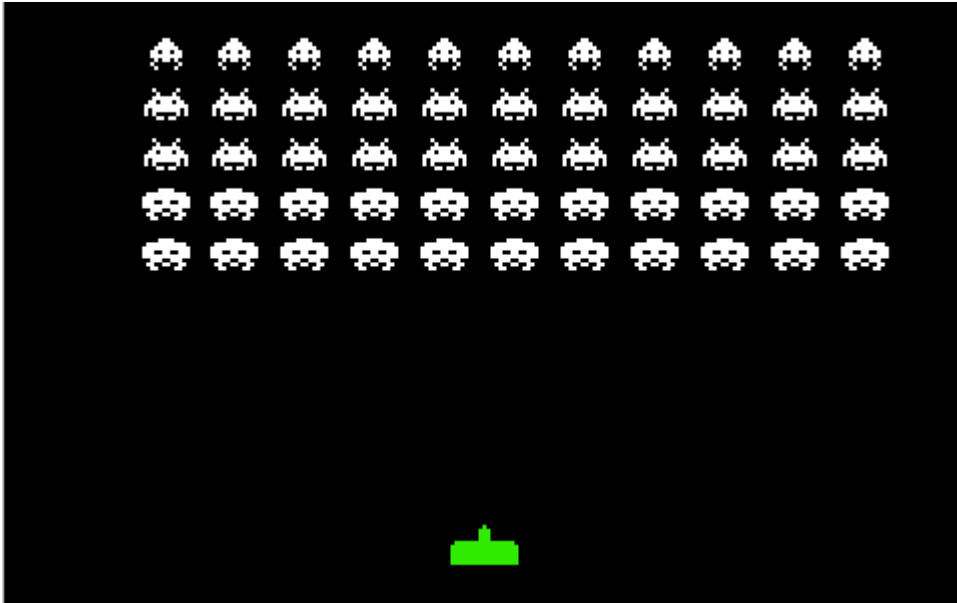
```
when I start as a clone
go to x: -175 + clone_counter * 35 y: -5
change clone_counter by 1
insert y position - laser_offset at clone_counter of fire_pos
insert 1 at clone_counter of alive_1
show
if clone_counter < 11 then
  create clone of myself
```

Now we just need to update the lists for the *refresh60* message to be *alive_1*, and change the points you get from 10 to 5. You should now have something that looks like this:

```

when I receive refresh60
  if boom = 1 then
    change explosion_counter by -1
    if explosion_counter = 0 then
      hide
      delete this clone
    else
      if alien_direction = -1 then
        change x by 0 - alien_speed
      if alien_direction = 0 then
        change y by 0 - alien_speed
        if y position < y position of tank + 25 and alive_1 contains 1 ? then
          broadcast game_over
      if alien_direction = 1 then
        change x by alien_speed
      if touching tank_laser ? then
        replace item clone_counter of alive_1 with 0
        change score by 5
        SetNewFirePos
        set explosion_counter to 8
        set boom to 1
        switch costume to explosion
  
```

And that's it! You now have 5 rows of alien invaders to fight off! If you start the game, they'll move around, and you can blow them up.



Adding the Mothership

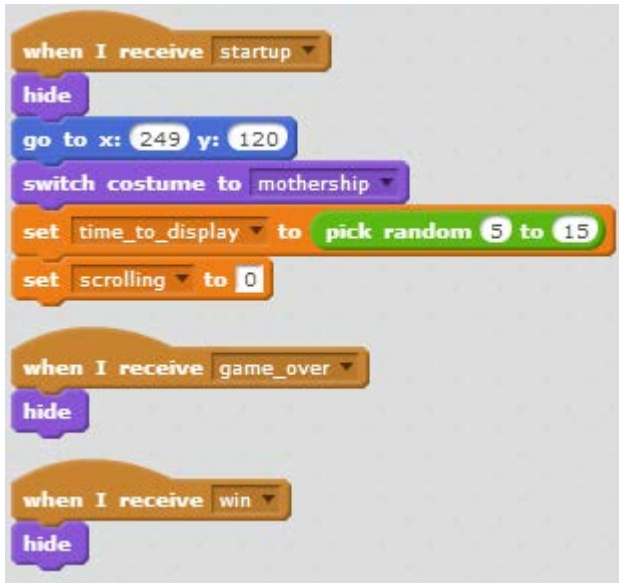
The one thing every alien invasion fleet needs is a mothership. We'll use the sprite we added way back when we added the tank laser.



We'll also need to add the explosion costume, so upload *explosion.png* now. The costumes should look like this:



The mothership will need a few scripts. First, we'll have the typical *startup*, *game_over*, and *win* scripts.



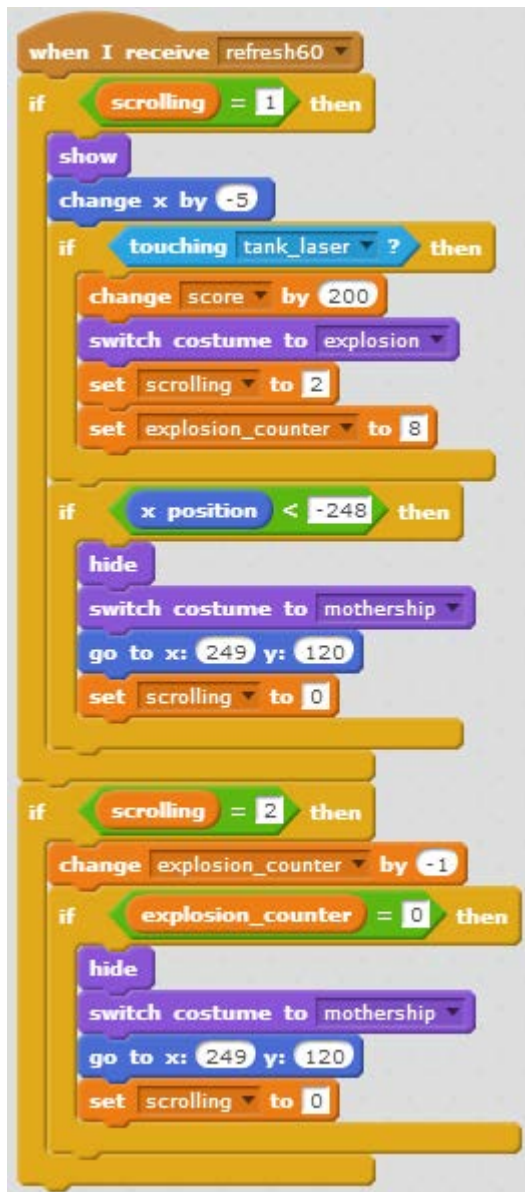
In *startup*, we'll hide the mothership and move it right to the edge of the screen, and make sure it's set to the non-explodey costume. We also set a couple of variables for this sprite only – *time_to_display* is the number of seconds until it flies across the top of the screen, and *scrolling* is the state of the mothership. Zero is waiting to fly across, one is flying across, and two is exploding.

The next script we have controls when it starts flying across. We'll use *refresh1* for this:



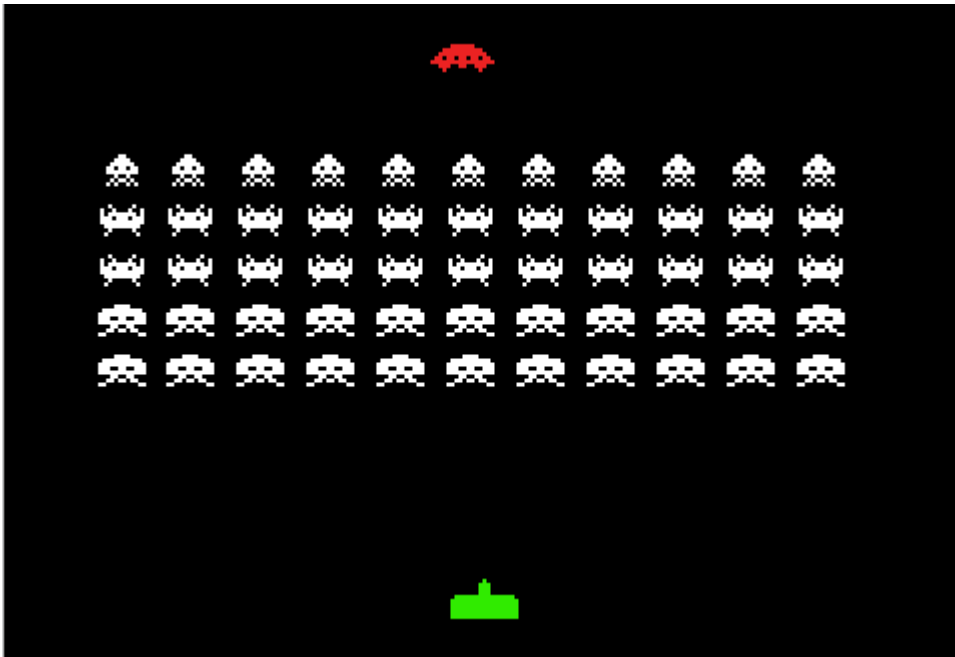
If the mothership is waiting to fly across, we'll check if the time to display has finished its countdown, and then set it scrolling, and pick a new time to display. Then, we'll decrease the time to display.

Now, we need to actually make it start flying across, using *refresh60*:



When the mothership is flying across (*scrolling* is one), we show the mothership, move it left by 5 pixels, and then do a few checks. If it is touching the tank laser, we need to make it explode, so we increase the score by 200, and change the costume to explosion. We also set *scrolling* to 2, so it doesn't try to keep moving, and set another new variable for this sprite only, *explosion_counter*, to eight. We also check if it has reached the left edge of the screen – if it has, we hide it, reset the costume, and move it back to the right side of the screen before setting *scrolling* back to zero, so it'll wait to go again.

If the mothership is exploding (*scrolling* is 2), we just decrease the explosion counter by one, and check if it has finished its countdown. If it has, we send it back to its initial position, and make it wait to go again.



Now that the alien fleet is complete, save. Next it's time to make it so that you can actually see all those points you're getting!

Keeping Score

Now that we have aliens to destroy, we need a way to keep track of it. We'll add a score counter for this. First, we want to upload a new sprite for the score label. This is the file *score.png*. Now we need to set up the scripts for the sprite. These are all pretty simple, because the score label just sits there.



When we startup the game, we'll just hide the label. Then, when the game starts, we move it to the top left corner and make it visible. And finally, when the game ends, either by winning or losing, we move the score down near the middle of the screen, and show it again, just in case something has hidden it.

Now we need to start adding the score itself. First, upload the file *0.png* as a new sprite. Name it *ones*, then add *1.png*, *2.png*, *3.png*, *4.png*, *5.png*, *6.png*, *7.png*, *8.png*, and *9.png* as new costumes. Make sure to organize the costumes so they go from 0 to 9, like below. Nine is just off screen below eight:



This sprite will handle the ones place of the score. First, we'll create the normal *startup*, *start*, *game_over*, and *win* messages:



When we run the startup, we'll first hide the number, and then move it to the right place. We also set a new variable for all sprites to zero, which will store the ones digit that we use. We then show the digit when we start. When the game ends, we move the sprite to an appropriate position along with the score label.

Next, we add the code that updates the number and sets the right costume.



When we receive the *refresh60* message, we'll set the variable to the score modulus 10. This is the remainder we would have if we divide the score by 10, which is effectively the ones place. Then, we switch the costume to that number. We have to add one, because the costumes are **one-indexed**, just like lists. This means that the first costume is stored at the position numbered 1. If it were **zero-indexed**, the first costume would be stored at the position 0.

Now that we've got the ones digit finished, we need to make the rest. Duplicate *ones* and rename the duplicate *tens*. We first need to change a few numbers in the startup and end game conditions, to shift the tens digit to the left by 12 pixels. The new scripts should look like this:

```

when I receive startup
hide
go to x: -112 y: 155
set tens to 0

when I receive game_over
go to x: 50 y: -25
show

when I receive win
go to x: 50 y: -25
show

```

We'll change the x positions for each of these scripts to the above values. We also add a new variable for all sprites, called *tens*. Next, we need to change the digit calculation, because we're working with the tens digit now:

```

when I receive refresh60
set tens to score mod 100 - ones
switch costume to round tens / 10 + 1

```

First, we need to calculate the tens value. To get this, we use *score* modulus 100 to give us the last two digits, and then subtract the ones value. This gives us the tens value. We then divide this by ten and add one to get the index of the costume we need.

Next, duplicate the tens sprite and name the new one *hundreds*. Add a new variable for all sprites, also called *hundreds*. Now, we need to modify the scripts.

```

when I receive startup
hide
go to x: -124 y: 155
set hundreds to 0

when I receive game_over
go to x: 38 y: -25
show

when I receive win
go to x: 38 y: -25
show

```

We update the x-positions, and change the variable we're setting to *hundreds*.



Then, we get the last three digits of the score, and subtract *tens* and *ones* from that so we only have the hundreds value. That can be divided by 100 and increased by one to give us the right costume.

Duplicate this sprite, rename the new one to *thousands*, and make a new variable for all sprites of the same name. We'll go through the same process again:



We'll update the x values, and set *thousands* to zero. Then, we'll move on to the *refresh60* script.



This time we'll get the last four digits, by taking *score* modules 10,000. We then subtract the hundreds, tens, and ones values, and set *thousands* to the remaining amount. We then divide that by 1000 and add one to get the index of the costume we want.

Duplicate this sprite, and rename the new one to *ten_thousands*, and create a variable for all sprites of the same name. Update the x-values and the variables in the scripts for the new sprite:


```

when I receive startup
hide
go to x: -148 y: 155
set ten_thousands to 0

when I receive game_over
go to x: 14 y: -25
show

when I receive win
go to x: 14 y: -25
show

```

Then, we'll update the code to set *ten_thousands*.

```

when I receive refresh60
set ten_thousands to (score mod 100000 - thousands - hundreds - tens - ones)
switch costume to round (ten_thousands / 10000) + 1

```

We'll use the same process of taking the score modulus 100000, and then subtracting all the remaining digits so we just have the ten thousands place. Then, we divide *ten_thousands* by 10,000 and add one to get the costume index.

Duplicate the sprite one last time, and name the new one *hundred_thousands*. Make a new variable for all sprites of the same name, and then update the startup and end scripts.

```

when I receive startup
hide
go to x: -160 y: 155
set hundred_thousands to 0

when I receive game_over
go to x: 2 y: -25
show

when I receive win
go to x: 2 y: -25
show

```

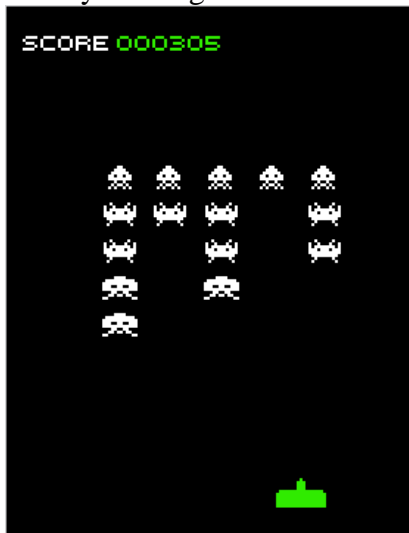
We'll update the x-positions and the variable before moving on to the *refresh60* script.

```

when I receive refresh60
  set hundred_thousands to score mod 1000000 - ten_thousands - thousands - hundreds - tens - ones
  switch costume to round hundred_thousands / 100000 + 1

```

Once we calculate *hundred_thousands* and its costume, we're finished up with the score. Before you move on, make sure to save. Try playing the game now, and see how high of a score you can get!



But you'll notice that they aren't shooting! Or if they are, the shots are just staying in one spot! We have to add the code to handle their lasers now.

Aliens That Fight Back

Switch over to the *alien_laser_0* sprite. The first thing we'll do is make sure the lasers handle startup correctly, as well as disappear properly when the game ends. You'll make three blocks of code responding to messages – *startup*, *game_over*, and *win*.

```

when I receive startup
  switch costume to alien_laser_0_0
  set costume_counter to 5
  hide

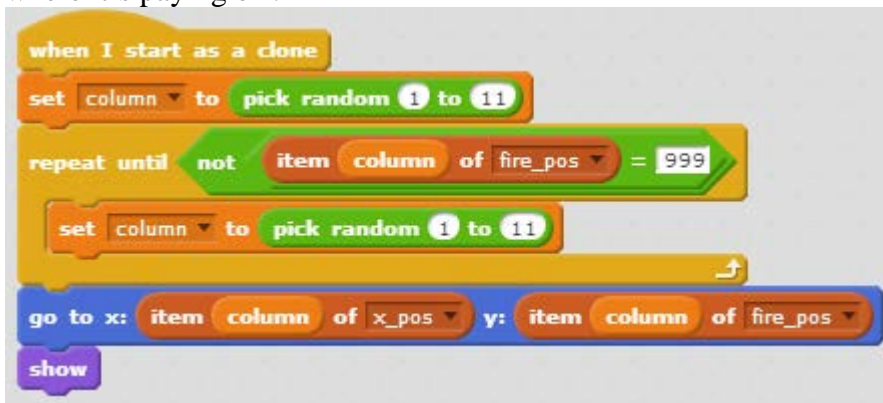
when I receive game_over
  delete this clone

when I receive win
  delete this clone

```

When the game sends the *startup* message, we'll set the costume to the first one, and set a new local variable, *costume_counter*. This controls how many *refresh60* messages occur before the laser switches to the next costume. Then, drag each of these scripts onto the *alien_laser_1* sprite. That sprite uses the same scripts, so this way we don't have to write them again.

Next, we'll add the script that runs when a new clone is created. If you remember back at the beginning, we added a script that created a clone of the laser at random intervals. This is where it's paying off.



First off, we set another local variable, *column*, to a random number 1 through 11. Then, we check if the value at that index in *fire_pos* is 999, meaning that that column shouldn't be used. If it is 999, we keep picking a new number until it isn't. Once we've found a column where it isn't 999, we'll go to the column's x position, and the height along the column that *fire_pos* gives us, and then show the laser. Drag and drop it onto *alien_laser_1* as well. But this still won't move or change costumes yet. That's where the *refresh60* message comes in. We'll add the following block of code:

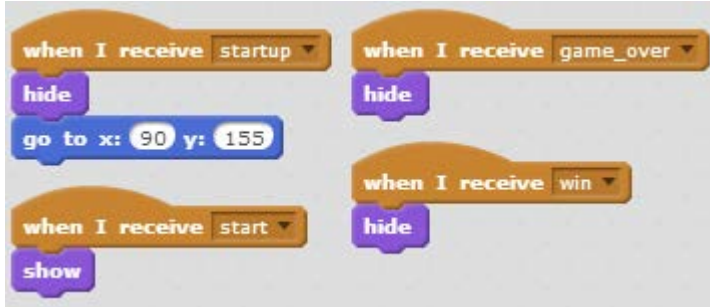


First, we check if the laser has reached the tank or the edge. If it has, we'll delete the clone, and taking off a life will be handled by the tank. If it hasn't, we move the laser down another ten pixels. Next, we'll check if the costume counter has reached zero. If it has, we reset it to 5, and then cycle to the next costume. Finally, we decrease the costume counter. That finishes off the first laser. Now copy this one over to *alien_laser_1* too.

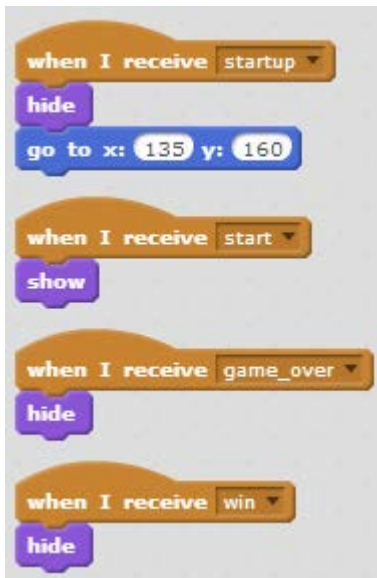
If we go to *alien_laser_1*, you'll see all the blocks are piled on top of each other. Spread them out, or right click and select "clean up" so Scratch does it for you. We're almost finished here, but you might notice that the "set costume to" block in *startup* is still set to *alien_laser_0_0*. Click on the drop down and change it to the first option, *alien_laser_1_0*. Now your aliens will shoot two type of lasers, and if you are hit three times, the game'll end.

Seeing Lives

Now we want to be able to see how many lives we have left. This'll require a couple more sprites. Make two more sprites by uploading *lives.png* and *tank.png*. Rename the new sprite made with *tank.png* to *life_1*. Now, open up lives, and add the following scripts for *startup*, *start*, *game_over*, and *win*.



Then we move on to *life_1*. Here' we'll need to add scripts for *startup*, *start*, *game_over* and *win* as well:



We also need to add a script for *die* as well:



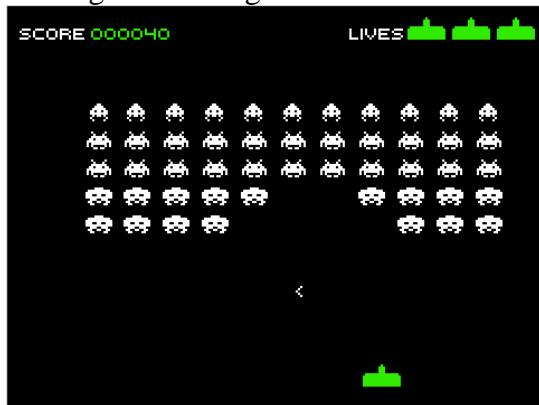
This will hide the sprite when you lose your last spare life. Now, we can use this same code for the displays for the other two lives. Duplicate *life_1* twice to make *life_2* and *life_3*. For *life_2*, we'll update the x-position in *startup* to 175, and change the number in the *die* script to two, so that the sprite will disappear when there is only one spare life left.



Now we need to do the same with *life_3*. We'll update the x-position to 215, and the value in *die* to 3.



Now as you get hit by the alien's lasers, you'll see your lives disappearing. But we're still missing a win and game over screen.



The End

We'll add two more sprites to finish off the game, using *victory.png* and *game_over.png*. First, we'll handle game over. Add these scripts to the game over sprite:

```
when I receive startup
hide
go to x: 0 y: 0

when I receive game_over
show
```

The scripts for winning are almost the same, so add those to *victory*:

```
when I receive startup
hide
go to x: 0 y: 0

when I receive win
show
```

Now, if you win or lose, the game will show you the appropriate screen, and your score!



Next Steps

Congratulations, you've created an awesome and complex game! From here, you can try to make the game better. Maybe tweak the point values, to make things more interesting, or change the alien's speed to make it more difficult. Or maybe add more levels (you should be able to do that by making some different messages, and using those for scripts similar to those for the *startup* and *start*)!