Bridging the Gap

Building Better Tools for Game Development

Daryl Tay (30449294)

This thesis is submitted for the degree of Honours in Multimedia in Interaction Digital Design in Murdoch University.

July 2010

Declaration

I declare that this thesis is my own account of my research and contains as its main content work which has not been previously submitted for a degree at any tertiary educational institution.

[illegible]

Daryl Tay

Copyright Acknowledgement

Signed: [illegible]

Full Name of Degree: Honours in Multimedia in Interaction Digital Design...

Thesis Title: Bridging the Gap – Building better tools for Game Development

Author: Daryl Tay

Year: 2010

Abstract

The following thesis is about questioning how we design game making tools, and how developers may build easier tools to use. It is about the highlighting the inadequacies of current game making programs as well as introducing Goal-Oriented Design as a possible solution. It is also about the processes of digital product development, and reflecting on the necessity for both design and development methods to work cohesively for meaningful results. Interaction Design is in essence the abstracting of key relations that matter to the contextual environment. The result of attempting to tie the Interaction Design principles, Game Design issues together with Software Development practices has led to the production of the User-Centred game engine, PlayBoard.

Acknowledgements

I would like to thank the following people, whom without their support, guidance and feedback would have never made this project even remotely accomplishable.

- My supervisor and Program Chair of Digital Media, Mark Cypher, for seeing me through the entire study and helping me refine, work and solidify the ideas throughout the course of study. Also for exposing me to the broad field of visual design and interaction which has deeply changed the way I perceive the world.
- My parents, for putting up with me, and allowing me to pursue this field of study, in being overseas more than a year, and also for their heart-warming concerns and attempts to communicate with me.
- My girlfriend, Nicole, for not only accepting me in my constant lack of time, but for always bringing an optimistic view into our lives.
- Charlie, the single person whom I have shared the most ideas about games, game culture and design with, for always listening to me and giving me the confidence to tackle any issue in regards to design.
- For everyone who has helped to test my program, as well as the encouraging feedbacks and useful criticisms that have helped to polish PlayBoard to the state it is now.

Contents

Introduction

"The less time you spend trying to figure out an application, the more time you can spend appreciating or using it. This is especially true in leisure applications where you are not in the mood to learn and find rather than explore and discover." Najib, IT Consultant.

Amateur game makers lead particularly distinct lives. They are individual artists who are able to realise their fantasies by crafting worlds, deciding interactions and playing in them. My own encounters with personal game making has given me the opportunity to witness the experience of playing my first and very own game, make games about my friends, make games for them, share in a community and receive many enthusiastic responses in return. Game making has allowed me to convey privatised scenarios, emotions, internal jokes and relish in localised topics that professional game makers would never even know about. It has allowed me to actually create my own deeply user-centred game experience, rather than rely on external and disconnected usability tests to craft one for me. It is through this appreciation of game making and its personal benefits that I wish to provide more opportunities for others to do so.

Computer game making is often perceived to be a lot more complicated than it should be. Professional game tools tend to project their functionalities, complexities and capabilities before allowing users a chance to understand basic usage; thereby alienating common users from effectively using them. After many conversations with game players it became clear that nearly everyone has game ideas to share, and a great many are capable of designing good games. However, those who are interested are often not able to digitally manifest their game ideas due to the lack of understandable and easy to use tools.

The lack of practical and good resources for indie game makers became the user needs I wanted to target. I wish to enable more user-centred game making tools to exist, and also to improve on their communicational qualities. This led to the decision to apply professional methodologies of Interaction Design to the non-professional field of game making through the production of a user-centred game making tool, eventually entitled PlayBoard. In the course of my research and presentation, I aim firstly to introduce the activity of game making to the basic user, secondly provide amateur game makers a better experience in making games, and thirdly highlight to existing developers the need to better our products through design.

Interaction design is in essence, a method of controlling products to distinctly target user objectives. Interaction Design acknowledges that subjective concerns can, or *must*, be treated as the professional goal in order to achieve any measurable form of success. My education in Interaction

Design has instilled in me the responsibilities of the producer and helped me to identify the needs of users in their various forms. It has also taught me the science of product refinement, which when applied appropriately according to desired contexts, *makes things better*.

I intend to use the knowledge of Interaction Design in the building of an amateur-targeted game making tool. In particular, I will be using the methodology of Goal-Oriented Design. However, Bridging the gap is by no means a goal-oriented project more than a pursuit-oriented statement. It recognises that there is a gap between communication of user needs and program developers, but it also understands that the magnitude of this gap is larger than one thesis, one man or a defined deadline. Game creation tools by convention will be built within a collaborative team, and shared with communities of contributors, artists, coders and designers who, in their own time, will bring a product to life. Nevertheless, it is about taking steps back, revisiting our values and restarting a different journey for the sake of realistic, substantial and relevant improvement.

As such, my first chapter will begin by describing the current game making and game playing environments, as well as how the interactions between the two have ultimately formed a distinct model of a community. This will be followed with the argument that having better tools will enable this community to grow. This chapter will also identify the major issues with current interaction design approaches that impede on game making and game playing cycle of interactions, and propose a design approach that simply places beginner needs before professional needs.

The second chapter will look into current game making interfaces from a visual design perspective. It will examine the two popular game products of usage, Unity 2 and Adobe Flash CS4, which are both recognised as progressive identities rather than fixed products. (eg. Adobe Flash CS4 will be replaced by Adobe Flash CS5 very soon) It will highlight some of the key areas in which interface designs have negatively impacted usage, as well as demonstrate how certain specific approaches are comparatively better. An analysis of interface details will highlight an underlying trend of design neglect that is evident in professional interfaces.

The third chapter compares and contrasts Model-Based Design and Human-Centred Design as approaches to design. Model-Based Design is about the efficiency, effectiveness and direct control of a product whilst Human-Centred Design concentrates on the engagement of the product with the user. Both approaches have to led to certain conflicts in the production process, but the examination of their differences will lead us to the conclusion that neither can be excluded in a production process as they are mutually dependent on each other to work.

Finally, we will look into Goal-Oriented Design, and how Goal-Oriented Design is an approach that best embodies the ideals and practical needs of software development. It will be compared and contrasted with a similar form of cognitive design, User-Centred Design, in terms of its practicalities, approaches and difference in production value. The result of these comparisons concludes that Goal-Oriented Design is a reliable methodology and was thus applied in the design and development of my program, PlayBoard.

A note on terminology

I would like to clarify some of the terminology I have used throughout the thesis. To my knowledge, there have been little official definitions of the product in which I call *game tool.* As such*, t*he identity of which I reference as *game tool* may also be interchangeably replaced with *game engine*, *game making platform*, *game making program, or game creational tool,* all of which are used to describe a program that supports a user in the creation of a game in any part of the constructional aspect.

Another class of terms I will refer to will be the *casual game developers*. I may also sometimes substitute these references with *independent game developers, indie game developers, casual game makers*, *amateur game makers* or any phrase which is similar in effect. These likewise refer to the same groups of people who would involve themselves in the process of game making for the sake of personal interest and non-commercial reasons.

This thesis has been referenced using the Chicago style.

Chapter 1. The User Needs

"However what we liked about Unity Indie was that it allowed many many people to get started with Unity. These people are hobbyists, students, professional and amateur independent developers, as well as teenagers and kids. And many of them are really valuable to the community." David Helgason on why Unity Indie was redistributed as a free product

User-centred interaction design philosophies begin with coming to terms with the user environment. (Cooper, Reimann and Cronin 2007, 13) In relation to game development tools, this environment not only refers to the virtual and physical spaces in which game producers and the players meet but also the communities in which they interact. Recent sociological studies reveal that user-centred ideals have not only contributed greatly in aligning the motivations, cultures and interests of users with those of the developers, but also the games they make and in turn the tools they use. The following chapter thus examines the game development scene and the impact the surrounding culture has on the many aspects of game creation. This chapter begins by introducing the independent non-professional game developer, as well as how they differ from conventional commercial game makers. The thesis will then argue that there is a significant and largely unsatisfied demand for amateur-targeted game development tools based on the relations and trends of these societal behaviours. Finally, the chapter will end with a proposed production approach that could be used to better target these community needs.

Game development has conventionally been regarded as an industry in which only specialised programmers, designers and businessmen have access and control. However, the widespread availability of game engines, middleware, and other content-creational tools has concurrently given the common user a chance to build their own games without the same level of expertise, networks, resource or even funding. (Fulp and Baez 2005, 28) The provision of developmental tools such as Adobe Flash and Unity has attributed to the rise of a new kind of game maker, the non-professional but nonetheless passionate amateur. Independent developers, though varying greatly in terms of skills and knowledge, are able to make use of the simplified process of game creation, to create and distribute personally-designed games. These game makers though largely unpaid are nonetheless valued for the amount of ideas, experiences and expressions they bring into their respective communities.

Avenues such as Newgrounds.com, Kongregate.com, MochiMedia, forums and personal websites amongst many others provide the virtual, social and environmental spaces in which independent developers meet and showcase their works. This is done through a variety of methods such as allowing developers to upload self-made products, connecting people of common interests together, providing downloadable tools and sometimes even giving out rewards to outstanding productions. Newgrounds is a web portal that hosts a large variety of Flash-based games and other assorted media, with little or negligible restrictions over production content or quality. Of the many hosting sites available, Newgrounds alone receives an estimated 50 user-submitted productions daily and has over

178,000 works to this date. (Newgrounds 2010b) Kongregate and MochiMedia, on the other hand, have an active community of experts and marketers who provide services and tips for amateur game makers who want to be published and advertised. Even game tool businesses such as Unity have begun to recognise the significant interest in game making which has led to their decision of converting their US$199 product, "Unity Indie", to be redistributed as completely free. (Helgason 2009)

Directly linked to the game producer is the common gamer and viewer that form the bulk of the community in game-hosting sites. Online gaming statistics indicate that there are about 134.5 million gamers who play online games in 2007, 42% of whom are female. The highest ranked genres of games were online card, puzzle arcade and words games (44%), family-oriented games (25%) and RPG/MMOGs (19%). (GRABStats.com 2008) Whilst Newgrounds may receive an average of 50-user productions each day, the people who play or view these productions total to an estimate of half a million daily. (Sheffield 2009, 21) These numbers represent the greater online society in which game developers contribute to and develop reciprocal relationships, either through further distribution, commenting, reviewing, responding or simply viewing. (Bruns 2006, 21) These numbers also serve as significant and profound evidence for the wide variety of interests in the field of gaming content, even when there is no differentiation between professional or amateur games. The most popular game on Newgrounds, is Hentai~ SimGirls (beta), which has individually received close to 44.9 million views to date, and was made by a singular author simply known as sim-man. (Newgrounds 2010a) For a figurative form of reference, the number of people involved in online gaming exceeds the entire population of Oceania and Australia combined, non-Internet users included, which stands at an estimate of 34.7 million in 2009.

The combined population of producers and game players make up an entire inter-dependent community. Independent game developers produce and share their works, in which casual players play, review and learn from; some of whom, in turn become independent game makers themselves and end up contributing back to the cycle of production and consumption. (Figure 2) This chain of interactive processes and interaction leads to a dynamic system of intercreativity, a fundamentally distinct model of user behaviour known as produsage. (Bruns 2006, 16) According to Axel Bruns, the author of "Blogs, Wikipedia, Second Life, and Beyond", the empowerments of technologies and its surrounding cultural environments has attributed to the rising trend of users turning into produsers. (Bruns 2006, 11) As referenced from other widely known phenomena such as Youtube, Wikipedia and blogs, users are evidently interested in playing active roles in communities as they participate in a range of activities. These include uploading custom-made videos on Youtube, writing personal reviews in their own blogs, and rating and commenting on current and new games. It is in this way that users contribute and sustain the knowledge of online culture. These various contributions help to support a thriving open community that provides both intrinsic rewards in production as well as external rewards of recognition. Independent game production, like its produser-centred relatives, also benefits from a culture of information sharing, collaboration and personal development within their own communities.

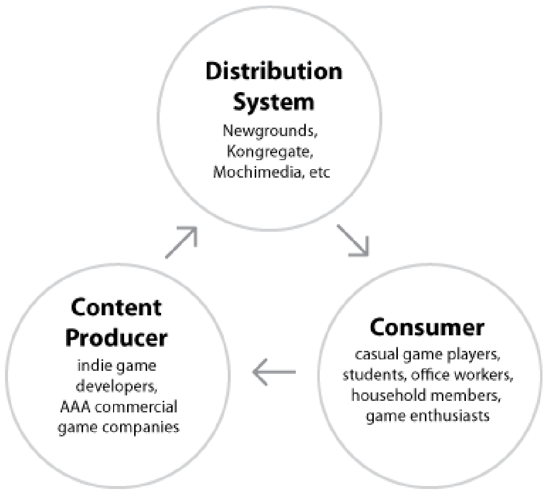

Distribution System
Newgrounds, Kongregate, Mochimedia, etc
Consumer
casual game players, students, office workers, household members, game enthusiasts
Content Producer
indie game developers, AAA commercial game companies

Figure 2.0 The self-sustaining system of intercreativity

As users evolve within a unique system of communal sharing to become produsers, their needs and demands concurrently shift and change. In a produser-centred environment such as amateur game development, the ideal function of a game tool should support the produser in the act of personalisation and appreciation of creative work. However, developing such a tool is notably difficult as it requires a substantial amount of technical knowledge, resources, time and understanding of usage. Game making is a largely expert-dominated industry. Hence the kinds of tools which are accessible, such as middleware, 3D modelling programs, physics engines and others, is in some ways still exclusive to experts only. This differs from produser platforms such as Youtube, Wikipedia and blogs, which are arguably made popular due to the easy-to-understand interfaces and low technical knowledge requirements. Following Cooper, Reimann and Cronin's graph (Figure3) of user proficiencies and given the assumption that game engines are still largely expert dominated, we can deduce a probability that there is a significant portion of people unable to cross the gap of difficulty. (Cooper, Reimann and Cronin 2007, 42)

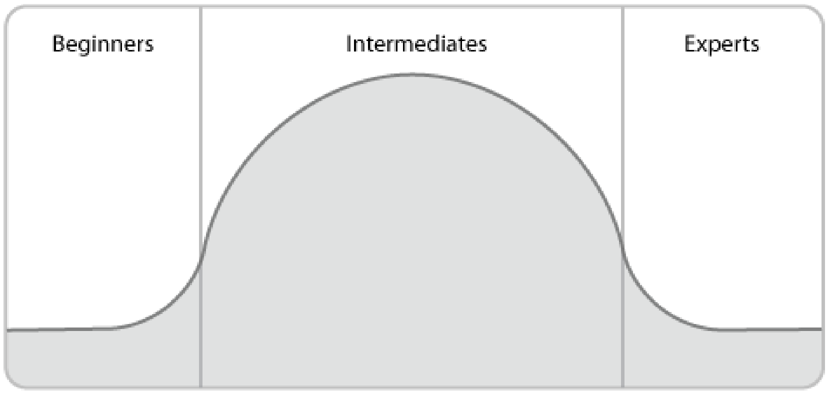


Figure 3.0 Graph depicting the normal distribution of user proficiencies in digital tools.

Many software developers are aware of their products being difficult to use. But when faced with the challenge of developing a game engine, they often choose to prioritise functionalities over usage. In response to a question whether Pushbutton[1] Engine would ever be possible for beginners to use, Jeff Tunnell, says, "We need to make sure PBE works for experienced coders first so we know the foundation is solid. Once we have that we absolutely intend for it to work for beginners." (Tunnell 2009) This may be a reasonable response and it shows signs of good intentions, but rarely does this mentality ever substantially benefit the common user. As referenced from other back-end focused communities such as SourceForge.net, programs by nature frequently fail to reach a state of full maturity, and are much less ever distributed or even heard of, or being used to such an extent that engages with general users in a revolutionary way. (SourceForge.net 2010) Beginner needs, being placed secondary, are thus highly unlikely to ever be satisfied, and has proven to rarely be met, especially when placed before the obstacles of expert needs.

Focusing too early on advanced capabilities leads to a large component of the target users being excluded from the development process. (Figure 4.1) It is perhaps better to consider the alternative, the initialising of a project that covers fundamental interactions before progressing further. (Figure 4.2) In doing so, advanced functions will have a better contextual placement, as they are built on top of relevant interactions, effectively still enabling the product to cater to expert needs.

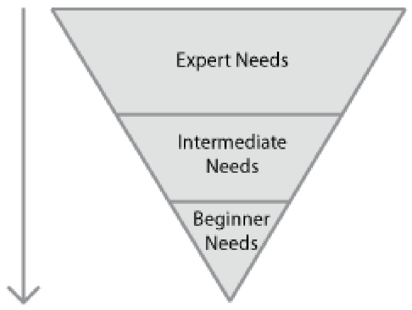


Figure 4.1 Conventional evolution of complex tools takes a long time before engaging with users.

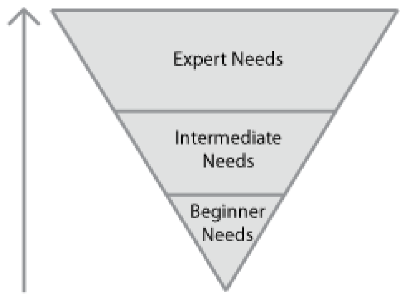


Figure 4.2 Suggested model of production. Involves having a basic interactive framework before adding advanced and possibly undesired functionality.

Independent game developers make up a large component of the intermediate and beginner users in most game-based communities. They contribute, through their own self interest, to areas of recognition such as online virtual spaces and in turn generate interest in other members of the community as well. A high level of interest within a field increases the likeliness of more developers being actively engaged and contributing to the growth of a community. This process of growth and interaction, although common in many other online networks, is restrained in game making communities due to the high levels of technical requirements involved. Judging from recent comments

[1] Pushbutton is an open-sourced game making engine that is used to support the Flash/Flex game development. It is built by veteran game engine developers, who have also been known to provide Torque, a 3D game engine. For more information, refer to http://pushbuttonengine.com.

of some makers of game tools this state of restrained development is unlikely to change due to the conventions instilled by linear development methodologies which place basic needs of users as secondary. Therefore, a re-examination of the design process of game making tools most relevant to independent game makers is necessary.

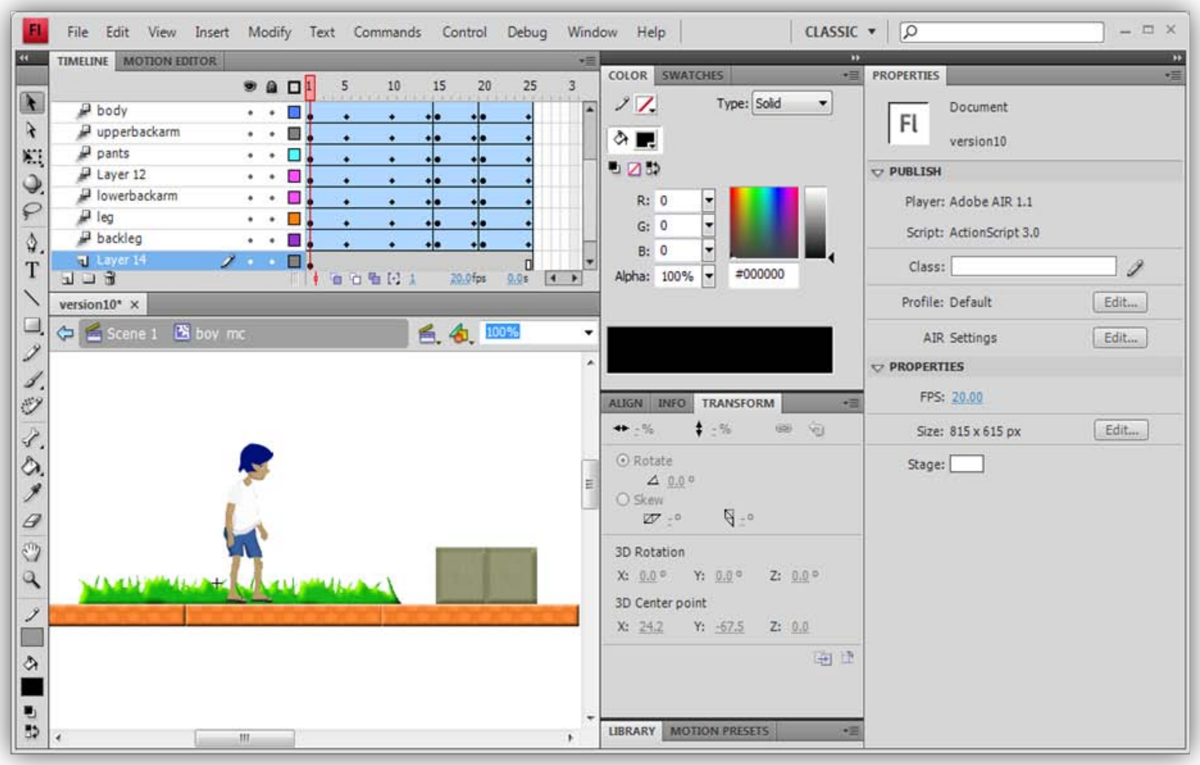
Chapter 2. The Evidence of Gaps in Current Interfaces

"If you ever think Flash is difficult to use, you should try drawing with a joystick on an Apple II before the concept of undo was invented." ~ Jonathan Gay, Creator of Flash

Visual interfaces are the primary and direct means by which a user encounters a product. Virtually all use stems from the understanding of the interface, which makes it a critical aspect to look into when designing beginner-targeted programs. Donald Norman, a cognitive psychologist, wrote about the many ways in which users derive affordances (possible action) from the visual impression of an artefact. (Norman 1988) The following chapter will reflect upon the various ways in which game-making interfaces have insufficiently communicated their appropriate affordances through visual presentation. In particular I will be looking at the interfaces of Adobe Flash CS4 and Unity 2 as they represent some of the more commonly recognised and frequently used independent game making platforms of today.

Adobe Flash CS4, released in 2008, is an instance of a game making application which, like its predecessors, provides a graphical interface for performing tasks like drawing vector images, positioning objects, managing symbols, and controlling object properties. It also allows users to write their own scripts through a scripting language known as ActionScript. Flash is not solely targeted for the development of games, as it is used widely in areas such as advertising, animation and websites. However, with 99% of current internet-enabled desktops supporting Adobe Flash Players and a plethora of sites showcasing professional and amateur Flash games, Adobe's Flash series is arguably the most widely recognised platform to address in terms of casual game production. Flash is in many ways a remarkable product in terms of its functionalities and its popularity, and it has undergone many iterations of design and modification, but still, as we shall soon see, its problems are numerous.

Upon initial observation, Flash's interface is notably congested. As seen in Figure 5.0, Adobe Flash CS4's interface features a considerably high number of buttons and windows compressed in a single screen. Donald Norman writes about how the number of controls provided should be relevant to usage, as the difficulty of performing a desired action exponentially increases with the number of irrelevant options provided. (Norman 1988, 208-209) With over seventy buttons sprayed from left to right and top to bottom within each panel (excluding the top bar and its timeline which are made up of an infinite number of frame buttons) and many unlabelled relationships and groupings, the amount of cognitive work required to use this product is exceptionally high. This design visibly tells the user that there are many things to do and many things to figure out, even if he or she is only interested in performing one task at a time. The presentation of Flash's interface is not only inconsiderate of the user's sense of perceived affordance; it is also likely to be daunting to the unfamiliar and untrained eye.



Type: Solid
R: 0
G: 0
B: 0
Alpha: 100%
#000000
Document: version10
Player: Adobe AIR 1.1
Script: ActionScript 3.0
Profile: Default
FPS: 20.00
Size: 815 x 615 px

Figure 5.0 A screenshot of the entire Adobe Flash CS4 screen in Classic Mode.

Moreover, key behaviours such as visibility of objects and relationships, current state of program and access to scripting are either hidden on first viewing, reduced in visual emphasis or simply given precedence to more temporal options. The accumulation of all these aspects makes the entire program significantly harder than necessary. As a natural part of usage, relevant information pertaining to actions should be made visible. (Norman 1988, 99-104) The Scene Editing stage area (bottom left) in figure 5.0 contradicts this notion by its significantly small size. The entire Scene Editing stage only takes up a quarter of the screen even though it is meant to reflect a full screen's worth of content. Even basic selection has many constraints applied to it such as being unable to select an object due to different "layer" properties. (Top left) On the other hand, the properties panel on the extreme right is given gratuitous space at its minimum width, thus reducing the emphasis on the main screen even more.

In addition, some items provided within the interface are of low practical significance, arguably even adverse to usage. The ability to retract windows is one such instance where a seemingly good intention can work conversely against the user. With a single click on the top bar, as represented in figure 6.1, an abrupt change occurs without consideration of how surrounding objects are shifted and affected. The top bar is a highly confusing behavioural object because firstly the icon does not resemble a conventional image of a button, yet is easy to click on by being excessively wide. (figure 6.1) Norman writes about the negativity of forcing functions, which describes designs that forces users to perform actions they do not intend to do. (Norman 1988, 132 – 140) Donald Norman also states how users should not be required to exercise high precision when using an object (Norman 1988, 58-59). Yet as shown in figure 6.2, this is not practiced at all. Small content tabs are placed close to the wide forcing button with no gap in between, increasing the risk of accidental clicking. In the instance where users want to select tabs rather than close them, this is counter-productive.

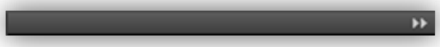


Figure 6.1 Button used for contracting and expanding panels

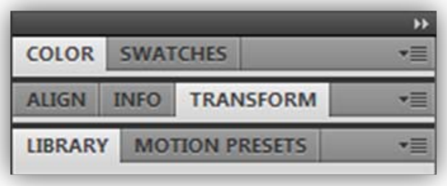


Figure 6.2 Example state of expanded panel

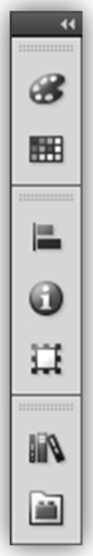
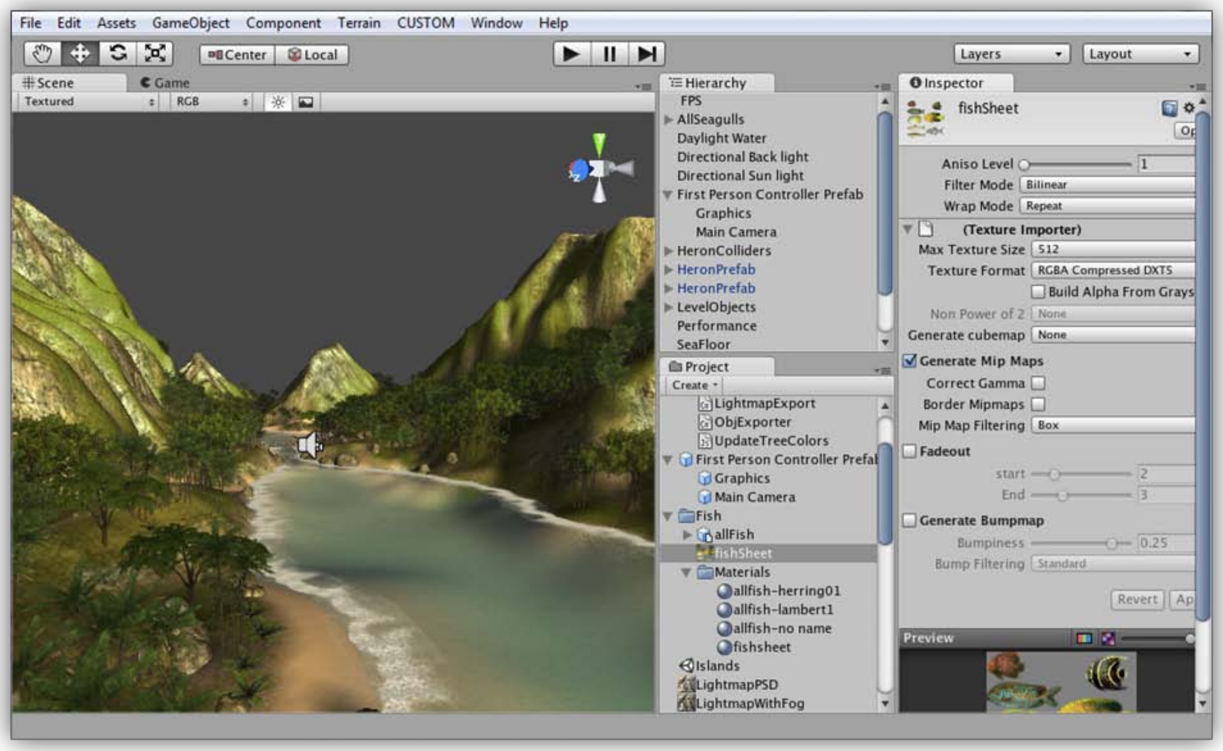


Figure 6.3
State of panel after accidental clicking. Demonstrating little visual relation with its expanded form.

Flash has numerous other problems in regards to its interface design but it is not my objective to list them all. Rather, the purpose of highlighting the negative factors of the interface is not to debate about whether Adobe Flash CS4 is a good or bad product but to show that gaps of understanding can and frequently exist, even in professional interface design. The next example depicts an interface from Unity 2 that has taken user needs into consideration, but even then contains several issues despite the attempts to make things easier.

Unity 2 is a specialised game making program that is largely used by professionals and independent game makers to build games for various platforms such as the web, Wii and standalone applications. It features many utilities and functions such as 3D object handling, shading and lighting, ease of access to external graphic programs and scripts. The functionalities of Unity are not targeted for beginners, but in many ways it is easier to understand than Flash in terms of its interface.

In comparison to Flash, Unity's interface is arguably easier to read. Although it features many visual elements like Flash, it makes use of spatial management to create visual order, making it easier to understand. Figure 7.0 illustrates Unity's undistractedly clear groupings which decreases the cognitive effort in finding and relating groups of objects. Furthermore, detailed objects are displayed in a consistent manner, all flowing in a downwards direction, alleviating the burden of interpretation and navigation. Words are also used more frequently to describe detailed properties and behavioural control rather than depending on ambiguous icons. In the context of using an interface, words are less likely to be misinterpreted, more able to represent unfamiliar behaviours and require no memorisation of meaning.



Aniso Level 1
Filter Mode Bilinear
Wrap Mode Repeat
(Texture Importer)
Max Texture Size 512
Texture Format RGBA Compressed DXT5
Build Alpha From Grays
Non Power of 2 None
Generate cubemap None
Generate Mip Maps ✓
Correct Gamma
Border Mipmaps
Mip Map Filtering Box
Fadeout
start 2
End 3
Generate Bumpmap
Bumpiness 0.25
Bump Filtering Standard

Fig 7.0 Screenshot of Unity 2. Illustrates the scene (main WYSIYYG working area) on the left, the object list in the middle and the object inspector on the right.

Unity is also significantly more directive simply because the options they provide relate visibly to achievable actions. Objects that are meant to be clicked on are all presented in a form of a bright button, slightly 3-dimensional, with a soft shadow on the edges to create contrast. The combination of these properties provides visual emphasis and makes interactive elements 'look more clickable'. In addition, key options are generally given gratuitous space in all surrounding directions to create distinction and thus demonstrate importance in comparison to the other elements. There are no toolbars for drawing, colour panels or abstract timelines that impedes on the display as they are not directly related to the game making process. This does not mean that such functionalities do not exist; they are simply hidden away until relevant to usage, allowing appropriate control over the user's desired actions. (Norman 1988, 208-209) Unity simply makes use of contextual placement and allows actions to be derived through relationships rather than from direct presentation.

The few panels provided by Unity's interface, though bearing some setbacks, generally support the user in quickly getting used to the navigational environment. This is a result of good conceptual modelling. (Norman 1988, 12) A good conceptual model is often a simplified representation of a commonly known complex structure that bears some comparable characteristics as its familiar form. Unity's "Hierarchy" and "Project" (figure 8.1) panel serve straightforward and predictable purposes as they behave like folders to provide lists of objects. However, the word "Hierarchy" could be more appropriately named "Used Objects" or any relatable alternative as its current name does not seemingly relate well with its function. The "Inspector" panel (figure 8.2), on the other hand, is well-named and behaves as it suggests. It dynamically reacts to specific selected objects, revealing every intricate detail upon inspection. As such users are then immediately able to control and edit their properties, as the intent of inspecting ties in relationally well with the intent to edit.

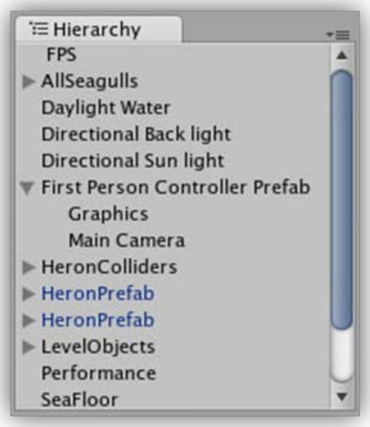
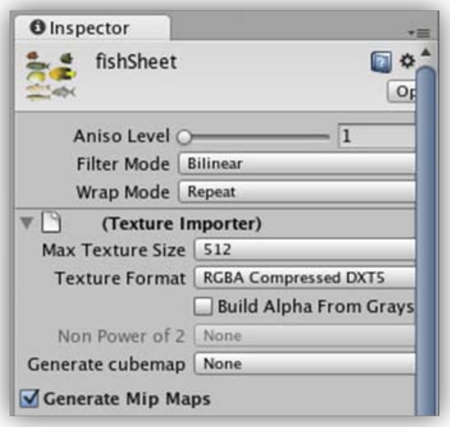


Figure 8.1 Unity 2's Hierarchy Panel behaving like a folder.



Aniso Level 1
Filter Mode Bilinear
Wrap Mode Repeat
(Texture Importer)
Max Texture Size 512
Texture Format RGBA Compressed DXT5
☐ Build Alpha From Grays
Non Power of 2 None
Generate cubemap None
☑ Generate Mip Maps

Figure 8.2 Unity's Inspector Panel which gives an indepth explanation of selected objects

We have thus compared the different types of communicational issues between Adobe Flash CS4 and Unity through their visual interfaces. Using Adobe Flash CS4 is firstly made difficult through the overwhelming number of options in its presentational screen. This also impedes on understanding through the lack of emphasis on key objects and behaviours whilst placing focus on unnecessary details. In addition, Flash is inherently cumbersome because some buttons possess unexpected behaviours, which are also prone to accidental clicking. Unity, on the other hand, is somewhat easier to use due to the establishment and enforcement of visual order. It also makes use of visual emphasis and distinction to highlight key objects that are important to navigation and usage, allowing a more guided understanding of the functionalities of the interface. Unity finally makes use of better conceptual modelling, through careful choice of descriptive words to explain elements that are otherwise difficult to understand. Whilst visual interfaces are only a part of the many factors that make up the user experience, the issues described in this chapter provides us with evidence of problem areas in current interface design.

Chapter 3. The Conflicted View of Design Ideals

"A designer is an emerging synthesis of artist, inventor, mechanic, objective economist and evolutionary strategist." Richard Buckminster Fuller, architect, author, designer, inventor and futurist.

In examining the qualities of interfaces, we reveal a layer of awareness, of lack thereof, about design ideals. Whilst it may be argued that design ideals are subjective to individual interpretation, they all relate objectively to a design's final outcome. As a result, it is important to understand how the different design perspectives affect a product and its relevant production process. This chapter introduces the opposing interaction design philosophies Model-Based Design and Human-Centred Design by briefly describing their significant traits. It will then proceed to compare and contrast the strengths and weaknesses of each theory, leading to the conclusion that both Model-Based techniques and Human-Centred Ideals are both needed in a project.

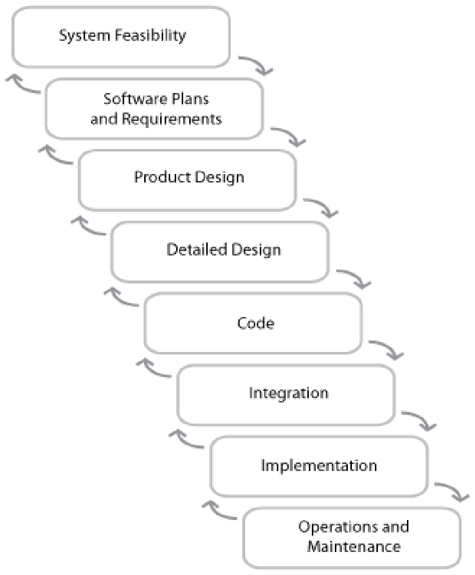


Figure 9.0 Waterfall structure of Software Engineering. One of the many examples of Model-Based Design approaches.

Model-Based Design focuses solely on an artefact's structural development. (Paterno 1999, 11) Examples of Model-Based Design include Object-Oriented Design, Bottom-Up Approach, Three-Level Architecture and the Waterfall Technique (Figure 9.0). These approaches tend to be managerial techniques that support the developer in the organisation and control of working environments. The attention to development is especially useful if the project is overbearingly large or consists of many confusing implemental details. (Paterno 1999, 1) It empowers the developer by providing a good mechanical and technical understanding of the entirety of the project, as well as the intricacies and relationships of the project elements. However, Model-Based Design is largely focused on the constructional issues of implementation. Therefore, usability, which is usually separated from the implementation stage, is often neglected or insufficiently dealt with.

Human-Centred Design, in this context, is thus the opposite of Model-Based Design; it has the user as its primary focus. Despite variable practical applications, Human-Centred Design overlaps with and indeed incorporates several other disciplines including User-Centred Design, Cognitive Design and Visual Design. The philosophy behind Human-Centred Design stems from the belief that products are essentially made for the user. In another words, easy to use products are essential for a product's success. (Norman 1988, 17-33) Human-Centred Design makes extensive iterations of observation, feedback, and modification to guarantee a continual improvement of product usage. Thus, information like how easily a product is understood (Norman 1988, 13), the number of items humanly remembered at a time (Norman 1988, 62-66) and the effect of visual objects on cognitive attention (Norman 1988, 14) all help to define the rules and guidelines for Human-Centred Design.

Comparatively, Model-Based Design supports all aspects of production. It breaks down the many stages of development into discernable segments which aids the developer in effectively managing, distributing or focusing on tasks. The clarity provided through Model-Based Design enables the identification of structural flaws, understanding of production affordance and also the likely expenditure of resource. Digital products especially tend to be technically challenging and often harbour many problems in the formulation, building and maintenance stages. Model-Based Design thus aids developers by enforcing a systematic workflow structure to minimise error and contain unpredictability. Disciplined methodologies have many benefits, such as enabling products to be built in less time, conserving the use of resources in production, enabling better capabilities in handling errors or simply being able to control a product's functionality. As Model-Based Design has been refined through the years of engineering and is a collection of working and tested techniques, it is thus highly useful and reliable in production.

Human-Centred Design, on the other hand, is valued because it pays attention to the user and user needs, albeit often at the expense of resource and time. In theory, an artefact's function is only as useful as the user perceives affordance and is able to actually learn how to use them; hence a product is greatly devalued if it does not communicate effectively with the user. (Norman 1988, 75) Modern interfaces especially tend to feature an exceptionally high number of accumulated functionalities, resulting in many opportunities for loss of interpretation and meaningful use. Without appropriate visual and behavioural feedback, all kinds of use-related problems can occur, from minor frustrations to severe and costly errors. Use-related problems exist in many forms, like being difficult to interact with, confusing to interpret, requiring an inappropriate skill level of the user or simply not being able to do what the user really wants. (Cooper, Reimann and Cronin 2007, 4-8) In more physical representations, errors in communication may be embodied as undesirable pop-up error messages, functions of gadgets that never get used or behavioural interfaces that have no reversible option. Although Human-Centred approaches do not aid in actually building products from scratch, it is valued because it tackles the most visible and possibly most important aspect; the ability of interfaces and software to correctly communicate its affordances with its user.

Yet, the role of Model-Based Design is just as evident in user experience as Human-Centred Design. Although rarely stated, the issues that stem from inadequate development processes are in essence also usability problems. Lack of functionality, unreliable behaviours or products that work too inefficiently are examples of usability issues that negatively impact the user. Harold Thimbleby, author of "Ignorance of Interaction Programming is Killing People", highlights the many ways in which poor practices in digital programming have deeply inconvenienced or harmed consumers, with instances as severe as accidental death. (Thimbleby 2008) Model-Based Design is perhaps the only way to reliably control the desired development of a product due to its ability to directly manipulate behaviour at a micro rather macro level. Therefore Model-Based Design supports the development process, which in turn contributes to the end user interactions, and makes Model-Based Design an intrinsic and fundamentally important part of the user experience.

However, simply ensuring working functionality is insufficient in guaranteeing a product's popularity. Human-Centred Design is still more suitable for enabling products to be better distributed and better appropriated to target audiences. By making products easy to interpret, more relevant and easier to use, Human-Centred artefacts are able to cater to a greater range of skill levels and interests. Whilst Human-Centred Design may have little influence over product development, it does allow a better control over how the product is used at a distribution level. The usability, aesthetics and visual communication of Human-Centred products work synonymously well with other areas such as marketing and advertising, allowing companies to target consumers who may not necessarily possess the expertise in the relevant field. "User-friendliness", for example, has become a promotional feature that has not only helped distribute products more widely but has made these same products more relevant to the average consumer. (Cooper, Reimann and Cronin 2007, xxix) On top of which, paying attention to user needs make people feel good, thus giving rise to branding and customer loyalty. Apple is one such company that till today benefits from a large following of customers because of excellent handling of user-related issues. (Cooper, Reimann and Cronin 2007, 4-8)

Whilst there are many other debates about the pros and cons of Model-Based Design and Human-Centred Design, ultimately what is required is not an explicit direction in either, but an incorporation of the two sides. A Human-Centred Designer cannot make a program more user-friendly if there is no working program to control; neither can a Model-Based Designer make a useful program for an environment if there is no knowledge of the environmental needs. The problem should not be about the weaknesses of either Model-Based Design or Human-Centred Design, but rather noting the lack of mutual understanding between developers and designers and end users. Projects in general will all have similar constraints, deadlines, target audiences, contextual information, resource limitations, user feedback handling, and product instabilities. Thus, both Model-Based Design and Human-Centred Design are necessary to handle all these variety of issues. Model-Based Designers can certainly benefit from learning user research techniques, and Human-Centred Designers in general would definitely benefit from learning to build products more efficiently and easily. A lack of any one of these fields can lead to a product failure, thus emphasising the point that no one aspect can be left out.

The uncontrolled issues that stem from poor execution of Model-Based Design invariably also become use-related problems. Likewise, Human-Centred Design has a greater influence over the users. Ultimately, both forms of design are deemed to be indispensible in a project. Despite having different approaches, Model-Based Design and Human-Centred Design are essentially complementary rather than conflicting.

Chapter 4. A Tool for Designing

“If you don't know where you are going, you will probably end up somewhere else.”
Lawrence J. Peter, Educator

Following from the understanding that both Model Based Design and Human Centred Design vary according to approaches, requirements and objectives, the question is still left unanswered as to what the ideal balance for an interaction design method might be. The imbalance between Model-Based Design and Human-Centred Design could be considered yet another form of a gap between designer and the ideal approach. In this chapter we shall thus introduce Goal-Oriented Design and how it can aid designers of game making applications in integrating the best of both Model-Based Design and Human-Centred Design. In comparison, we will be evaluating Goal-Oriented Design against User-Centred Design to demonstrate the difference in practicality and stability of the two approaches. Finally we will conclude that Goal-Directed Design is a more suitable interaction design methodology due to its coverage of design issues, depth of analysis and applicability to game making software.

Of the two contrasting positions between Model-Based Design and Human-Centred Design, Goal-Oriented Design has emerged as a hybrid of the two. Goal-Oriented Design was founded by Alan Cooper, Robert Reimann and Dave Cronin. It focuses on deciphering and handling goals related to a context and covers a step-by-step procedure for completing a project. (Cooper, Reimann and Cronin 2007, 24) It is an expanded version of Activity-Centred Design and also shares the same core values as User-Centred Design by focusing on user values. (Cooper, Reimann and Cronin 15) The entire process is made up of six iterative steps (Figure 10) which is understood as research, modelling, forming requirements, building framework, refining, and revising necessary details. (Cooper, Reimann and Cronin 2007, 20) In essence, the entire Goal-Oriented approach makes use of project goals to become an inherent part of the work process - helping to control the structure of activity, which in turn help to direct the nature of tasks. (Norman 2005, 16) This process bears similarities to the practical benefits of Model-Based Design in the way it aids the designer in production, but also shares its root philosophies with Human-Centred Design’s concern for the user.

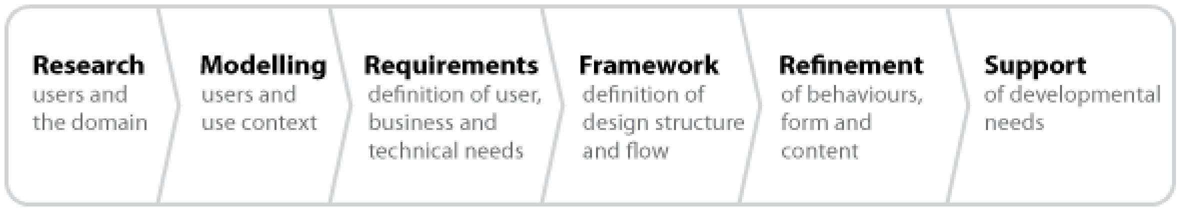

Research
users and the domain
Modelling
users and use context
Requirements
definition of user, business and technical needs
Framework
definition of design structure and flow
Refinement
of behaviours, form and content
Support
of developmental needs

Figure 10.0 An abstract depiction of the complete Goal-Oriented Design process. This will be used as the methodology for designing PlayBoard. However, as PlayBoard is a prototype, this thesis will only demonstrate the Goal-Oriented structure up to the point of “Framework”.

The limitation of User-Centred Design is that user testing is only effective at the end of a project, when a basic prototype is already built. (Cooper, Reimann and Cronin 2007, 70) User testing is the process of having a number of people attempt to use a product whilst identifying problems. However, from the developer's point of view, user testing is difficult to act on, particularly if the testing is done only towards the end of development. As the product is already largely defined, it becomes difficult to modify due to its technical nature and also due to limitations on time constraints. Likewise if problems become apparent in critical areas such as design structure or product purpose, the whole product may even be redesigned which is highly inefficient and generally economically unfavourable. Therefore, whilst usability tests may be an easy way to spot and edge out shallow mistakes, it does not cater well to deeper problems in a design throughout the development process.

In terms of effectiveness, User-Centred Design practices are often unreliable. (Cooper, Reimann and Cronin 2007, 71) Even though user testing is directed at seeking out and handling all errors, this process is flawed by the fact that common users are sometimes not capable of giving helpful feedback beyond what is obvious. (Alan, Cooper and Reimann 2007, 4) Despite their best intentions, what a user says may be misleadingly different from what he or she means and differences of opinion between users can also result in a loss of interpretation. General users lack the knowledge and familiarity of the product to adequately comment about all issues that the designer may be looking for. As a result, interpreters are forced to deal with inconsistent test results and worse still, depend on these statements to derive follow-up action. (Norman 2005, 17)

Goal-Oriented Design, like User-Centred Design, is a problem-targeting approach. Goal – Oriented design does this by dealing with user goals rather than simply handling usability mistakes toward the end of the production process. Goals are determined through the study of potential users before production. Research involves observing likely clients in their respective workplaces or homes and noting their traits, habits and preferences. (Cooper, Reimann and Cronin 2007, 50-70) The gathering of qualitative information helps designers to understand the product's contexts which in turn are used to formulate project goals. (Cooper, Reimann and Cronin 2007, 72-73) Prior research decreases misunderstandings between user and designer at early stages of development and it also brings into awareness the cultural and environmental issues relating to user needs. This allows the derivation of clear and relevant directions which enable the rest of the development crew to stay focused on key concerns; this is arguably necessary for work efficiency and effectiveness.

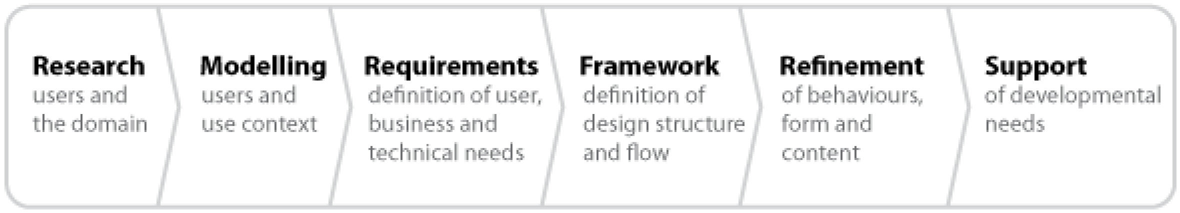
Goal-Oriented Design is likewise concerned about users, but it manages user issues more precisely by splitting user identities into specified groups and relating products to their contextual needs. Goal-Oriented Design makes use of an idea called "Personas" which are fictional references that signify example user characteristics. (Cooper, Reimann and Cronin 2007, 76) This usually involves the writing of an example background, an elaboration of user traits, naming the character and also providing a portrait to complete the image. (Figure 11) Personas are often created from the research data that is part of the Goal-Oriented Design process, and is meant to group related user concerns under categorised similarities. Cooper et al argues that having an identity to reference is very useful for designers, as it allows a representational concept to embody abstract interactive ideas such as intention, unsuitability and satisfaction.

Carefree Karen
On concerns with her career
- Does not wish to give up her personal life for work
- Wants to be in an environment in which she can nurture her skills
- Is happy with a medium-range level of payment
- Uncertain of the her job opportunities based on her degree

Figure 11.0 A picture being displayed on the left, and characteristics on the right. An example depiction of persona being used in a design process.

The greatest weakness of User-Centred Design is that it simply is not a complete methodology. Although User-Centred values are easy to understand, it lacks clear instructions on how a designer is supposed to create a product. (Cooper, Reimann and Cronin 2007, 9) User-Centred Design was written by a cognitive psychologist and not a designer. Therefore, a specialist from a different field is not expected to know the nuances involved in all designs as there are far too many possibilities to predict. In the context of an application, it is important to understand how to integrate design needs into the development process in order for the approach to be effective. User-Centred Design mentions little about the project structures and work flows that actually make up the design process. (Cooper, Reimann and Cronin 2007, 9) In this sense, User-Centred Design, though valued by many in the design industry, is essentially an incomplete approach that serves as a checklist rather than a complete production or design guideline.

On the other hand, Goal-Oriented Design has more coverage in terms of instruction. Unlike User-Centred Design which mostly states a list of constraints, Goal-Oriented Design actually goes through the step-by-step process (figure 12 on the next page) of how to design a product from the early stages such as research, to the planning of meetings and development, as well as the reiterative changes to guarantee completion and success. (Cooper, Reimann and Cronin 2007, 24) The coverage of instructions stated in Goal-Oriented Design extends to all aspects of production, including the designer's working context and client's environment, greatly increasing the level of feasibility and clarity. It scopes down large possibilities into several specific goals of achievements and follows up with recommended actions to achieve these targets. On top of which, Goal-Oriented Design will always stay relevant to the project because it is structured to function relative to the appropriate context.


Research users and the domain
Modelling users and use context
Requirements definition of user, business and technical needs
Framework definition of design structure and flow
Refinement of behaviours, form and content
Support of developmental needs

Research

Scope: define project goals and schedule
Audit: review existing work and product
Stakeholder Interviews: understand product vision and constraints
User interviews: understand user needs and behaviour

Scope	**Audit**	**Stakeholder Interviews**	**User Interviews**
Objectives, timelines, financial constraints, process, milestones	Business and marketing plans, branding strategy, market research, product portfolio plans, competitors, relevant technologies	Product vision, risks, opportunities, constraints, logistics, users	Users, potential users, behaviour, aptitudes, motivations, environments, tools, challenges

Modelling

Personas: user and customer archetypes
Other Models: represent domain factors beyond individual users and customers

Personas	**Other Models**
Pattern in user and customerbhaviours, attitudes, aptitudes, goals, environments, tools, challenges	Workflows among multiple people, environments, artifacts

Requirements and Definition

Context Scenarios: tell stories about ideal user experiences
Requirements: describe necessary capabilities of the product

Context Scenarios	**Requirements**
How the product fits into the personas life and environment and helps them achieve their goals	Functional and data needs, user mental models, design imperatives, product vision, business requirements, technology

Fig. 12.0 A detailed examination of the research, modelling and definition phases of Goal-Oriented Design.

In conclusion, Goal-Oriented Design is far more complete and therefore suitable as an approach for the questions of this thesis compared to User-Centred Design. User-Centred Design makes it difficult to implement changes when its main focus is toward the end of the design process. On the other hand, Goal-Oriented Design can be applied at the initial stage of research and makes use of contextual understanding to prevent problems from arising earlier in the process. Furthermore, Goal-Oriented Design understands its users throughout the production process better than User-Centred Design and more aptly manages user issues. User-Centred Design is considered an incomplete approach, lacking realistic instructions for actual application whilst Goal-Oriented Design is highly detailed and has an extensive coverage of instruction. For these reasons, Goal-Oriented Design is more developed and reliable, and hence more feasible for realistic use. To further demonstrate the notion, Goal-Oriented Design will be used as an adopted methodology toward the development of a prototype of a 2d game development platform, PlayBoard.

Chapter 5. Applying Goal Oriented Design

The following chapter documents the application of Goal-Oriented Design in the design and development of the amateur-targeted game-making tool, PlayBoard. As PlayBoard is only a prototype, meant to demonstrate an initial stage of design, the product did not complete all stages of the Goal-Oriented Design process. Instead the approach has been followed until the point of “Framework”, where further testing and refinement was intentionally uncovered. The building of Playboard is therefore documented through the Goal-Oriented processes of Research, Modelling, Requirements and Framework as depicted in Figure 13.

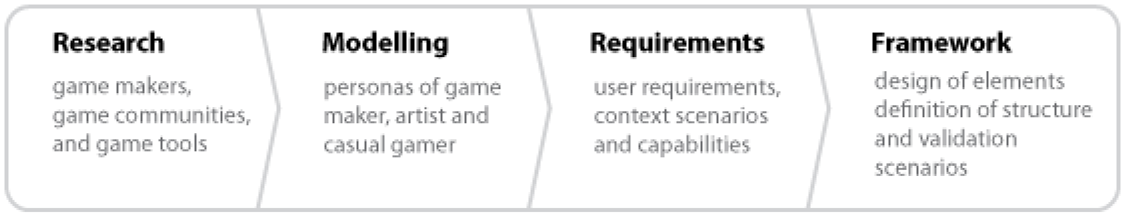

Research
game makers, game communities, and game tools
Modelling
personas of game maker, artist and casual gamer
Requirements
user requirements, context scenarios and capabilities
Framework
design of elements definition of structure and validation scenarios

Figure 13 The abstracted flow of Goal-Oriented Design used in the construction of PlayBoard.

5a. Research of Users and Domains

The research data in this section is about independent game makers and their surrounding environments. It summarises the concerns, issues and factors of the users PlayBoard is targeting. It is also directly used to create the user personas as demonstrated in the following chapter.

- Independent game developers are interested in a wide variety of topics, are generally engaged in expressive activities and often make games that reference popular identities as well.
- Their works are often judged by large active communities of game players, who are generally able to critique quality of games according to in depth detail, execution and overall appreciation, demonstrating a notable form of expertise in game content.
- Actual number of online game players makes up a third of online population. These are likely to be mixtures of casual gamers as well as more serious gamers.
- Independent Flash Developers refer to various forums and tutorials in order to learn how to script and use functions. These tutorials are unsorted and difficult to piece together without prior knowledge.
- Game players greatly outnumber game makers 10000 to 1, indicating a high probability that a large number of players are interested in making games but do not know how.
- Newgrounds, Kongregate and Mochi Media, all feature Flash-based games, and are very popular.
- Flash is installed on majority of all computers, making it highly accessible and conveniently distributable.
- Flash is commonly used, but its interface is not designed for the common user.

5b. Modelling the Personas

The three personas represent the target users of the program PlayBoard. These personas are entirely fictional, but are based on distinct characteristics that make up the diverse and probable user community. In particular, PlayBoard's broad primary targets are of the beginner-intermediate region, which in this case is represented by Simply Sandy and Expressive Ed.

Capable Colin

Colin is a full-time programmer who has plenty of experience with creating digital products. He is proficient in several coding languages, as well as the surrounding code conventions in professional production. In his free time he answers questions posted by Internet users on forums in regards to project developing strategies, technical difficulties and other related concerns. He also keeps up-to-date with open-source news and may help in the development of a project if he finds it exciting or interesting. Colin is a vital part of the game-making community as he plays the role of the advisory "expert".

- Colin is involved with an active community of game makers and is likely to want to share his games with them. Being able to publish standalone games would therefore be one of his main requirements for usage.
- Colin has more ambitious game making needs. He likes having a high level of functionality in programs, and expects certain kinds of conventions such as a level editing mechanic, physics engine and scripting capabilities.
- Colin has a large history of project files and his most desired trait of a game making tool is the ability to import his own assets, code and external files.
- Colin understands that not every new product comes out perfect immediately and takes time to refine, but prefers it if the product was open-source so he can edit it himself.

Expressive Eddie

Eddie is a full-time student who loves all forms of creative works. He frequents online communities such as Newgrounds, DeviantArt and Youtube and finds joy in the kinds of inspirations and interactive experiences his virtual peers provide. Having already dabbled in photography, web design and animation, Eddie wants to progress on to game design to share his many ideas. In terms of game making, Eddie would be considered a beginner, but he has a good wealth of prior experience in game mechanics to know what game making is likely to be about. Eddie is PlayBoard's primary target as he is likely to be a continuing user.

- Eddie is looking for the simplest and quickest way to effectively get his idea out. He is willing to put up with minor setbacks and difficulties but is ultimately impatient in 'getting things moving'. He detests unnecessary clicks and interruptions, and is more likely to ignore the reading of a manual.
- Eddies makes many mistakes in trying to get an idea out, he also changes his mind very frequently. He expects the program to be unaffected by constant change, as well as for the program to cater to backtracking and modification.
- Eddie is a fundamentally visual person. He makes many assumptions simply based on how an object looks, and does get affected or offended by concepts that do not look like they behave.
- Eddie is self-driven and is highly explorative. If he is given three basic colours, he will discover how to paint an elaborate picture.

Simply Sandy

Sandy is one of the many users who believe that technology is too difficult for her to use. Her daily life revolves around social activities and travelling, and rarely requires the need to make use of any digital tool. However, she has played several games casually, and is familiar with the concepts of gaming. Sandy has a good sense of fun, but when asked if she would make be interested in making games, she would reply, "If I can understand it!"

- Visual experience for Sandy is very important in attracting and maintaining her attention. As she is inexperienced with the different kinds of game making platforms, she wants to find something she can connect and identify with.
- Sandy has no experience in game design, so she needs a clear and quick visual reference to indicate understandable action.
- As Sandy is unfamiliar with the conventions of interfaces, and as such would require programs to allow her to learn and discovery at her own pace.
- Sandy naturally has very little resources in terms of graphics, sounds and code. She is also unfamiliar with the means to get them. Her ideal program would not require her to do so.

5c. User Requirements and Context Scenarios

From the personalities and preferences of the distinct users, we deduce the desired role of PlayBoard and relate it to the product design. For example, the development of interactions should always be focused on being efficient as Eddie is generally impatient, whilst the visual presentation of PlayBoard should be attractive to Sandy to help her maintain interest. In methodologically comparing and interpreting the individual traits of users, we derive with a set of user requirements. The table below represents the summary of the program's desired traits based on the personas described in the previous chapter.

User Requirements

Capable Colin	**Expressive Eddie**	**Simply Sandy**
Distributable	Efficient	Attractive
Capable	Forgiving	Directive
Flexible	Obvious	Safe
Open Sourced	Expansive	No prior requirements

However, simply having a list of desired traits does not complete the actual sense of usage. These traits have to be applied through a course of time, through meaningful interactions, and in regards to their contextual scenarios. A new program is not likely to be fully understood in an instant; its identity progressively changes according to usage, understanding and familiarity. Since PlayBoard is meant to introduce game making ideas to new users, the flow of information should be well controlled and predictable. In order to understand how users are likely to view PlayBoard, we thus simulate their responses according to predicted forms of usage. Understanding the different usage also helps to better envision user's temporal goals and which should also be taken into account in the program's design and development. The presentation on the next page demonstrates the context scenarios of Colin, Eddie and Sandy according to five different stages of usage - Learning, Experimentation, Familiarisation, Acknowledgement and Mastery.

<u>Context Scenarios</u>

Learning

“Okay, so how does *this* program work?” Colin

“This looks interesting.” Eddie

“What is this?” Sandy

Experimentation

“Ah, I see. That’s handy.” Colin

“Nice, it responds to me!” Eddie

“Oh, I did something. It’s prettier than I expected.” Sandy

Familiarisation

“I wonder if it works if I try doing this here instead.” Colin

“How did I do that again?” Eddie

“I didn’t know I could do that.” Sandy

Acknowledgement

“I can see where this is going.” Colin

“This is interesting.” Eddie

“That was easier than I expected.” Sandy

Mastery

“Now all this program needs is ...” Colin

“I am going to make a game out of this.” Eddie

“This is quite fun to play with.” Sandy

5d. Constructing the Framework

This section is a loose chronological summary of the development process of Playboard. It is an abstract of the events I felt best embodied the significant moments of adherence and deterrence of design goals. Whilst the goals of the project were largely fixed, there were many unexpected constructional issues that arose during the course of development, causing me to backtrack and reconsider new information in my approach. In many ways, modelling my artefact and experimenting with its behaviours before development was a form of research that occurred very frequently during the development process.[2] Technicalities and details in areas such coding which are not directly relatable to the purpose and argument of my thesis were left out. My documentation will explain the various ways I have struggled to achieve a holistic balance in consideration of the different requirements and needs.

The first aspect I had focused on was the formation of the visual cognitive model. The goals I had highlighted for the visual interfaces were to be supportive of high level interaction without impeding on vision. From the very start, I was looking for a presentational model that was easy to relate to game design. I immediately knew that I could not reference any of the popular game making platforms, firstly due to the fact that they were not commonly known, and secondly because of their lack of friendliness that made it unsuitable for Sandy or even Eddie-type users. However, in referencing the aspects that made these game platforms *functional*, I deduced the need to provide a stage, some form of navigation, options and a channel of feedback. I looked into various other utility-type conventions and found that most utilities used a constant grey-coloured interface. This I felt was important to retain as the game screen had the potential to be visually erratic. Grey is the most neutral colour and works like a background; essentially pushing the stage area and other active areas forward.

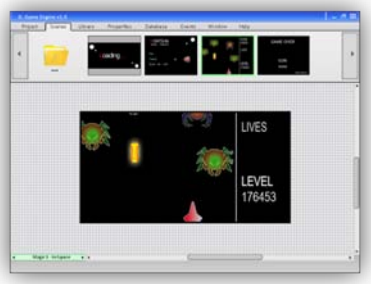


Figure 12.1 Microsoft Word Inspired Design

Knowing that I could not use many of the conventional game making interfaces for as a reliable reference, I looked for other utilitarian programs that the public was already familiar with. My first design was inspired by Microsoft Word 2007. (Figure 12.1) It had most of its functions at its top bar, as well as the use of tabs that helped symbolise areas of selection. It even featured a blue header bar to look like a Windows program and to balance the dull colours of the interface. However, after testing this proved to look *too much* like a utilitarian program, and would only relate to Colin-type users who did not mind staring at boring interfaces.

[2] PlayBoard was programmed in Adobe Flash CS4 with Adobe Air and Actionscript 3.0. This was due to the ease of installation and cross-platform capabilities that helped to reinforce the "user-friendliness" of the program. A desktop application, rather than a browser-based application was also decided on, so as to make use of localised files in future extensions.

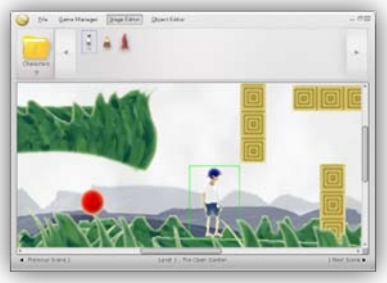


Figure 12.2 Stylised Utilitarian Design

This interface design was eventually reworked through several other iterations to become notably glossier, with various gradients and shiny edges to highlight buttons and tools. (Figure 12.2) Unfortunately, in focusing too much on graphical detail, I had missed out on a certain key area of usage. Informal comments on the visual presentation helped me to highlight that the presentation did not look like it was meant to do anything in particular. It was important to visually project the kinds of options a user would wish to know about.

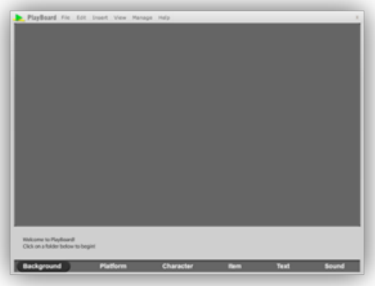
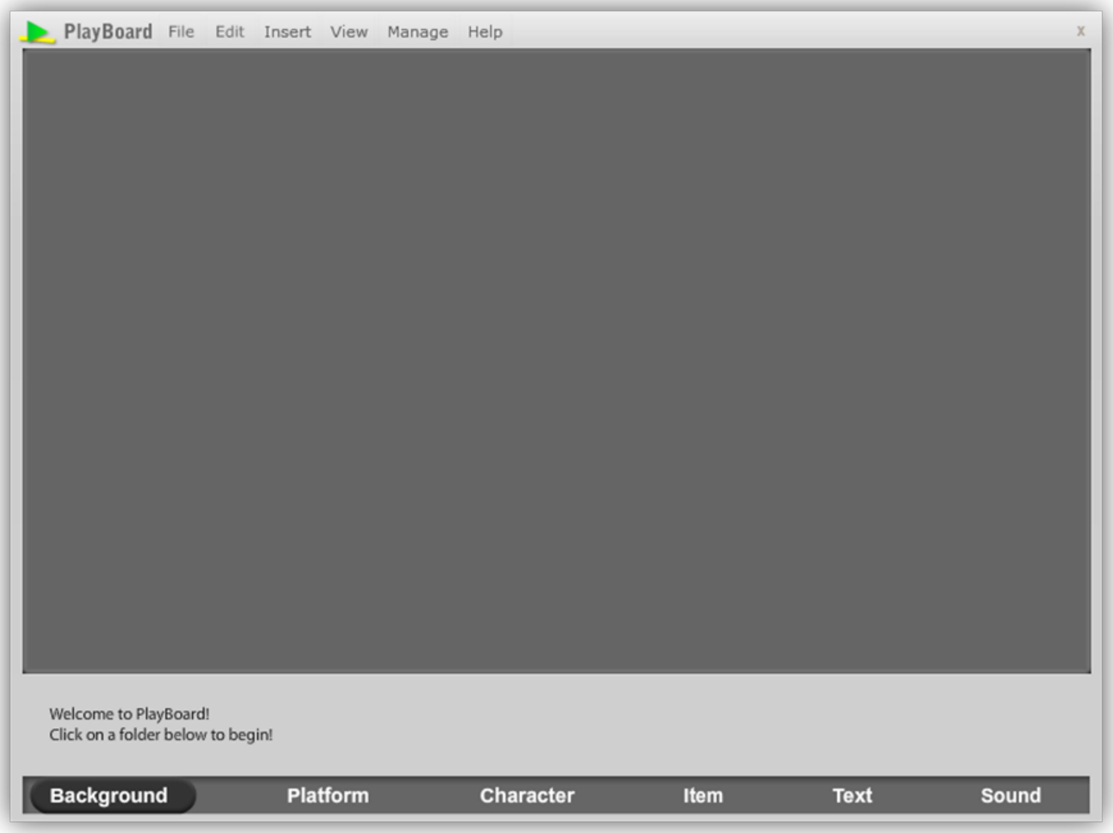


Figure 12.3 Minimalistic version of visual interface

This eventually led to the requestioning of what was truly needed in relation to matching user cognitive design with a basic interface. I knew that there were several factors to take into account, such as the interface should not be too colourful so as to distract the eye; it had to host a visual stage, selectable icons, some form of navigation and a feedback channel. Yet, at the same time it had to have some level of visual interest, whilst communicating the nature of its activities. What was not well answered was *how* much or to what degree I needed to accomplish all this, and how I was going balance these traits with my own constraints and resources. The only real measurement of standard I had was primarily my eye, and secondly the opinions of testers.

I therefore, started a process of simplification (Figure 12.3), taking out elements that were not absolutely necessary and prioritising the functionalities that were needed. I decided to leave visual polishes and attractiveness as a separate issue for future touch-ups. In the mean time, I removed every element that could afford to be removed. In doing so I was trying to take into account the personas requirements whilst imagining how the user would actually be interacting with the interface. After which, I placed essential elements back onto the stage, in a row by row manner, starting with the left and proceeding to the right - the conventional reading directions of western languages. I also used a hierarchy of functions of sorts, by placing, arguably most important elements in the top left and the least on the far right. This procedural placement was to provide as much familiarity of order (as reading order is commonly recognised, as well as the convention used in software tools) as possible.



Welcome to PlayBoard!
Click on a folder below to begin!

Figure 13 PlayBoard's current Interface design, demonstrating the spaciousness and cleanliness that derives from horizontal spatial usage and reinforcements of left to right order, as well as subtle elements described in Figure 15.

To my own surprise, the strict, methodological process of simplification produced a visual style by itself. It was minimalistic, and at the same time clear. The significance of each object and their individual relationships were recognised and the absence of elements such as scrollbars, buttons and labels diminished distractions for the eye. Minimising elements also allowed better control of widths and spaces, allowing me to create a sleek form of presentation that was incidentally aesthetically neat, without disrupting the order of presentation. In addition, I managed to create a large amount of space (Figure 13), which was a desirable trait for new users such as Eddie and Sandy who would be turned off easily by cramped or cluttered interfaces.

However, in shifting the menu bar down, the navigational area became significantly less prominent. The navigation bar (figure 14) is fundamentally linked to all aspects of usage, and it was thus important to not only be visually noticeable, but also had to reflect its behaviours. I made use of several text icons to solve this issue, primarily because they were horizontal images, which fit neatly within horizontal bars. To create attention on these texts I made use of contrast, spacing and size. Being an entirely gray-scale design, the only option of contrast I was logically able to use was of the distinction between white and black. White, being the more visually attractive colour was chosen as the font colour, which was also bolded for additional emphasis. The surrounding environments were then evenly and gratuitously spaced and shaded, and the texts were enlarged to make them look prominent.

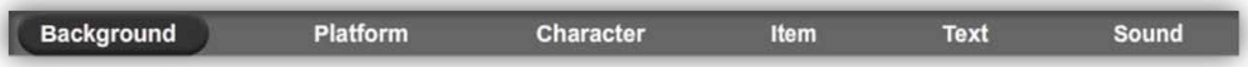


Figure 14 The navigational bar at bottom of stage, designed to be distinguished from the rest of the program

As the major areas of usage were laid down, the visual order naturally became self-evident. It became easier to fill these spaces up and knowing which areas to leave spacious because of the clear distinctions of groupings and relevancies. There were several other subtle details (figure 15) about the use of alignment, spacing, shading, typography and sizing that were put into place to ensure an entire coherency of presentation. As a final product, the visual presentation achieved a level of 'obviousness' that I felt would satisfy the Eddie type user.

Figure 15.1 Dynamically moving button which is very darkly coloured to provide contrast with navigational text. Is given a slight 3-dimensional look to its edges which helps bring a sense of forwardness and distinction to its contextual area, which was deemed to require visual attention. The button is also the only element in the interface which has a rounded corner, providing contrast in shape and therefore producing emphasis. Its flattened rectangular shape and protruding sides once again hints at the horizontal left-to-right reading flow.

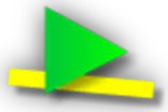


Figure 15.2 The PlayBoard logo, used to signify a direct representation of Play icon on a board. The slants and arrows are also meant to point towards the right, enforcing the overall reading direction, whilst the board hints downwards, indicating the importance of bottom areas. Green represents permissibility and yellow for humour, reflecting the personality of an easy-to-use program for beginners.

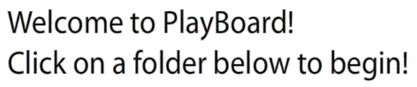


Figure 15.3 The PlayBoard prompting text. This was an area to provide subtle hints and feedback as the user used the program. As such, it was placed in contrasting bright area and centralised vertically to ensure readability. Its positioning was also close to the initial points of usage, the navigation and object elements, thus making it easier to read. Yet, its low positioning also meant it could be easily ignored for Colin and Eddie type users who do not wish to be guided when using the program.

Figure 15.4 PlayBoard, the name was chosen as a synonym for a physical property. PlayBoard establishes multiple relationships and metaphors of board games, still objects, getting on board, playing or an object with physical properties. The application's interface was also large, flat looking and resembled a board, thereby reinforcing contextual identity. The font chosen was Bell Gothic Std, a san serif font for cleaner visual impression, which had balanced boldness and clear spacings. With these two aspects combined the Bell Gothic Std best embodied a friendly and supportive look. The fonts colour was also close to black, but not of utmost darkness, to reduce the extremity of its impression, implying feelings of 'softness'.

Aesthetics

Aesthetics is an important part of the game making process because of the use preferences of the personas Expressive Eddy and Simple Sandy. These users are primarily sensorial rather than conceptual, and aesthetics was one of the key ways to provide users with a sense of gratification as the stage filled with personally placed objects. This also provided beginners like Sandy, who is not expected to have her own images or know how to create one, a sense of visually being able to build something quickly.

Following User-Centred Design's guide of making use of familiar conceptual models, I made an assumption that platform games such as Super Mario were well recognised and easy to identify with. As a result, I decided to provide graphical objects that all related to the common visual conventions of platform games. (Figure 16) I drew, and to some extent animated, a basic set of platforms, grass, blocks, characters and stars. The provision of graphics was not meant to control the actions of the user, but allow the user to form their own interpretations through the placement of them. Nevertheless, I provided objects that looked like they came from platform games to give beginners a starting idea of how these objects can be used.

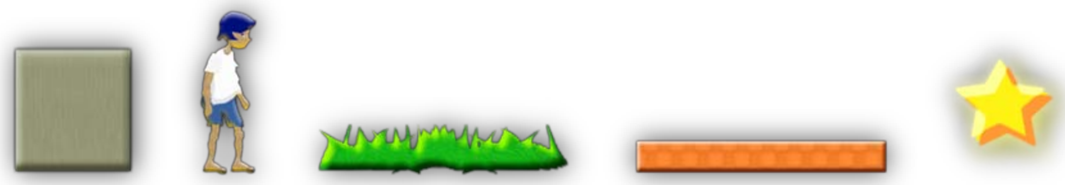


Figure 16 Graphical Objects which showcases a variety of bright colours, distinctive shapes and familiar platform game style representation of objects.

These objects were designed to contrast the dull-looking grey interface. As such they are all distinctly shaped, full coloured and to some extent textured, depending on the model of representation. All of these features aid in the creation of visual emphasis. They were also made to look like they popped up slightly from the screen providing a visual hint of their affordance; that is to be interacted with via clicking and dragging.

In addition to the visuals, I provided the options to include sounds, because they are important aesthetic elements and complete the entire game making experience. In this way I had only one sound file, which was personally edited from a previous game I had made, and I also provided a music clip. This music track[3] was composed by a friend named Luigi Alvarado in Costa Rica, and was used with his acknowledged permission. This form of file sharing in turn reflected the nature of amateur game communities, where artists, coders, designers, musicians and sound artists frequently share their works to combine into a multimedia object.

The interface was not complete without interaction. Upon implementing the individual item behaviours, there were many other unperfected areas that became evident through informal and personal testing. For example when attempting to navigate between folders, the transition of pages

[3] The track used is entitled "Daytime Detectives", by Luigi Alvarado (Composer for Media) – Costa Rica.

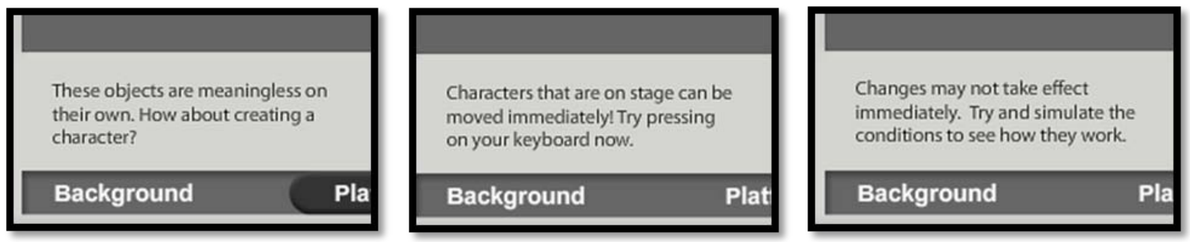
happened too suddenly, leading to unnecessary surprise. Abrupt reactions to mouse clicks or movements can cause a loss of spatial interpretation, which negatively disrupts the learning process. It became recognisably important to control the feeling of interaction, and to reduce its impression of miscommunication.

I made use of animation to give the user a sense of responsiveness that was natural and affirming. This was firstly done through the elastic motion of the navigational button that followed the mouse cursor wherever the user navigated to; as if the button reaches out and attempts to understand where the user wanted to go. The movement as well as spatial effect better reflected the progressive nature of navigation, reducing the potential feeling of a user getting lost.

Secondly, all usable objects highlighted themselves whenever being hovered, indicating a sense of pre-selection. Objects that were dynamically selectable would respond by lighting up with a white glow as the mouse hovered over them. This gave the user a basic recognition of what was being affected, and increased perceived affordance which became significantly more important as the game building process progressed and cluttered with objects.

Thirdly, game objects that were interactive made use of drag and drop mouse mechanics to be placed and moved onto the stage. Dragging provided a sense of manual control that was at the same time easy to perform, enforcing the idea of confident usage in Sandy and Eddie. It also helped to reduce the likelihood of occurrences of accidental clicking in unintended areas, or accidental movement when objects were already in place.

Closer to the end of the production process, when I had many visual objects in place, I was able to envision the relationships in the interface with the user significantly better. This allowed me to design an entire framework of predicted usage intentions, which I in turn used to develop a prompting system to guide the user in interface usage. This prompting system would read the user's current actions, and compare it with the user's history to provide a suggestion that would be appropriate to the respective context. (Figure 17) I felt this form of communication was much like a conversation in real-time rather than a documentation which had to be read linearly. This has noted to be one of the most significant factors which made the program easy to comprehend, as affirmed by my several testers.

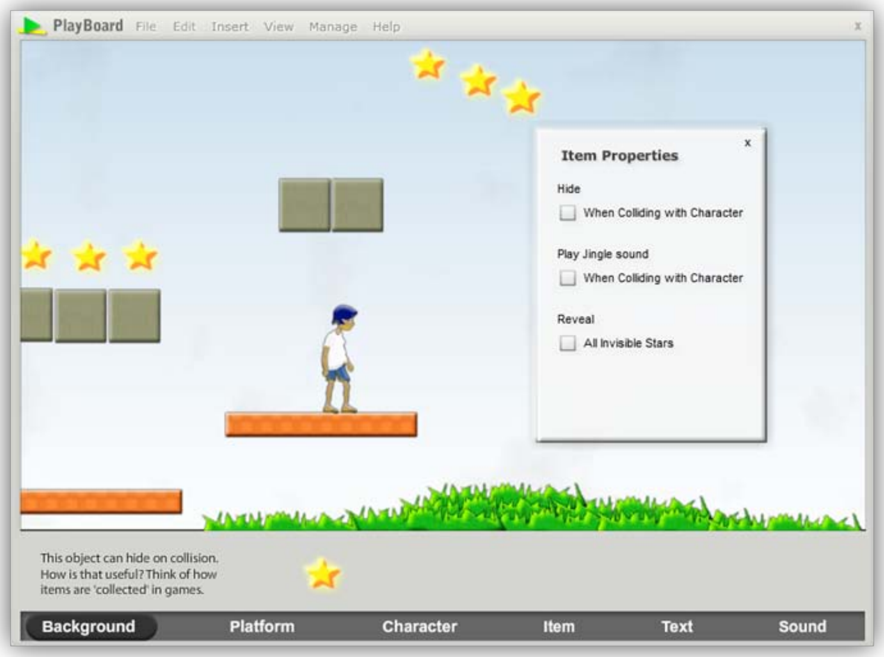

These objects are meaningless on their own. How about creating a character?

Characters that are on stage can be moved immediately! Try pressing on your keyboard now.

Changes may not take effect immediately. Try and simulate the conditions to see how they work.

Figure 17 Example messages that the prompt displays to aid user in the ease of learning.

Understanding the full cycle of usage also allowed for better reflection of user-desired tasks in the actual work process. I was able to design a framework that could to some extent distinguish when the user was interested in simply viewing the stage, as compared to when the user actually was looking for in-depth information. This allowed me to create a dynamically reactive window that provided appropriate options when the user was trying to examine an object. (Figure 18) Conversely the same window could also hide to conserve visual space when the user was not examining a game object. This function became a significant part of the interface use as it was relevant to all interactive objects within the application.



Item Properties

Hide
☐ When Colliding with Character

Play Jingle sound
☐ When Colliding with Character

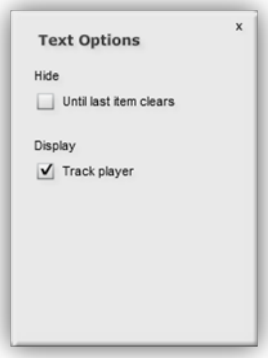
Reveal
☐ All Invisible Stars

This object can hide on collision. How is that useful? Think of how items are 'collected' in games.

Figure 18 The pop-up information window in the right central area that appears as user selects an object.

Lastly, I had to design a simplified process of game logic, so as to introduce the concept of game making to new users. I referred to my experiences in programming, and the use of the conceptual object commonly entitled "functions", and presented them as visual options to toggle on and off. (Figure 19)

These functions work by pre-coding the sets of behaviours that belonged to an object, and are activated only when the user manually turns them on. The functions would then visually appear during usage, whenever a user examines an object. In doing so, the user can personally control the effects of how each behaviour worked in regards to their contextual placement.



Text Options

Hide
☐ Until last item clears

Display
☑ Track player

Figure 19 A window depicting options of a game object's logical processes. The options were specially selected to help new users figure out game logic through controlled presentations.

I also made use of the fact that these options were only of two states, on and off, to effectively reduce the complexity of usage, even though it could still be used to create relatively complex effects.

This form of visual presentation became a useful way to represent the “public” and “private” properties of objects commonly understood in “object-oriented” coding conventions.

5e. Usability Testing

From testing the prototype with several users, there have been a fair variety of responses. Game development had not been an easy concept to introduce, but was met with relative amount of success. Users who firstly proclaimed that they were avid game players, all managed to figure out how to create their own stages as well as create a logical display of win conditions. Expert users, who had game-making backgrounds, were able to understand the intent of the interface and had commented with statements like “Very good program for beginners.” There were several true beginners who managed to perform the tasks as required, but admitted that they still did not understand fully how to use it. Most of the minor issues such as inappropriate prompting, or lack of indication, that have been highlighted during the course of testing have already been adopted into the program as fixes. However, PlayBoard is only a demonstration of a program’s potential state through the presentation of its prototype. PlayBoard is designed to continually expand and develop through time and as such, will require a continual effort to identify, control and communicate elements to satisfy user needs.

See Appendix C and Appendix *D* for the consent form and questionnaire provided to program testers. As the testing was concerned about collective opinions, actual responses are not published to retain confidentiality of identities of each individual.

Conclusion

Gaming today is not only a leisure activity, but a potential means of communicating experiences with each other. Each game possesses the capacity to create memories, inspire, relate and teach concepts at rates that exceeds many other traditional forms of media. A game is essentially an idea, embodied through representation and interaction. It is the recognition of the value of personalised ideas that I have attempted to provide users with a game creation tool.

In line with Goal-Oriented Design's first step to research users and their domains, the first chapter describes the current amateur game making communities made up of game makers, players and the environments in which they mutually interact. I have highlighted how the interactions between the members of these communities have resulted in a system of intercreativity, which is not only notably popular but were in part responsible for the generation of new game makers. I have also identified the interaction design approaches that impede on game making and game playing cycle of interactions, as well as proposed a design approach for game design tools to target these communal needs more effectively.

The second chapter examined two of the current game making platforms Adobe Flash CS4 and Unity 2 through their visual interfaces. It elaborated on how Adobe Flash CS4 negatively impacted usage by congesting too many elements within a screen, projected unclear relationships and provided functions that were not only impractical, but detrimental to usage. The chapter also explained how Unity 2's interface was comparatively better because of the enforcements of visual order, grouping and appropriate behavioural conventions that attributed to an easier learning experience.

The third chapter examined the traits of Model-Based Design and Human-Centred Design as representatives of opposing design philosophies. Model-Based Design was noted for its reliability in production and capability in tackling development-based issues, as well as its fundamental role in enabling user experience. Human-Centred Design, on the other hand, was valued for its attention on the user, ability to make objects meaningful, as well as enabling better distribution and marketing of products. Yet, upon examining the realistic and demanding needs of a design project, it was evident that neither Model-Based Design nor Human-Centred Design could be excluded from the design process, and that both the strengths of Model-Based Design and Human-Centred Design were necessary to create successful products.

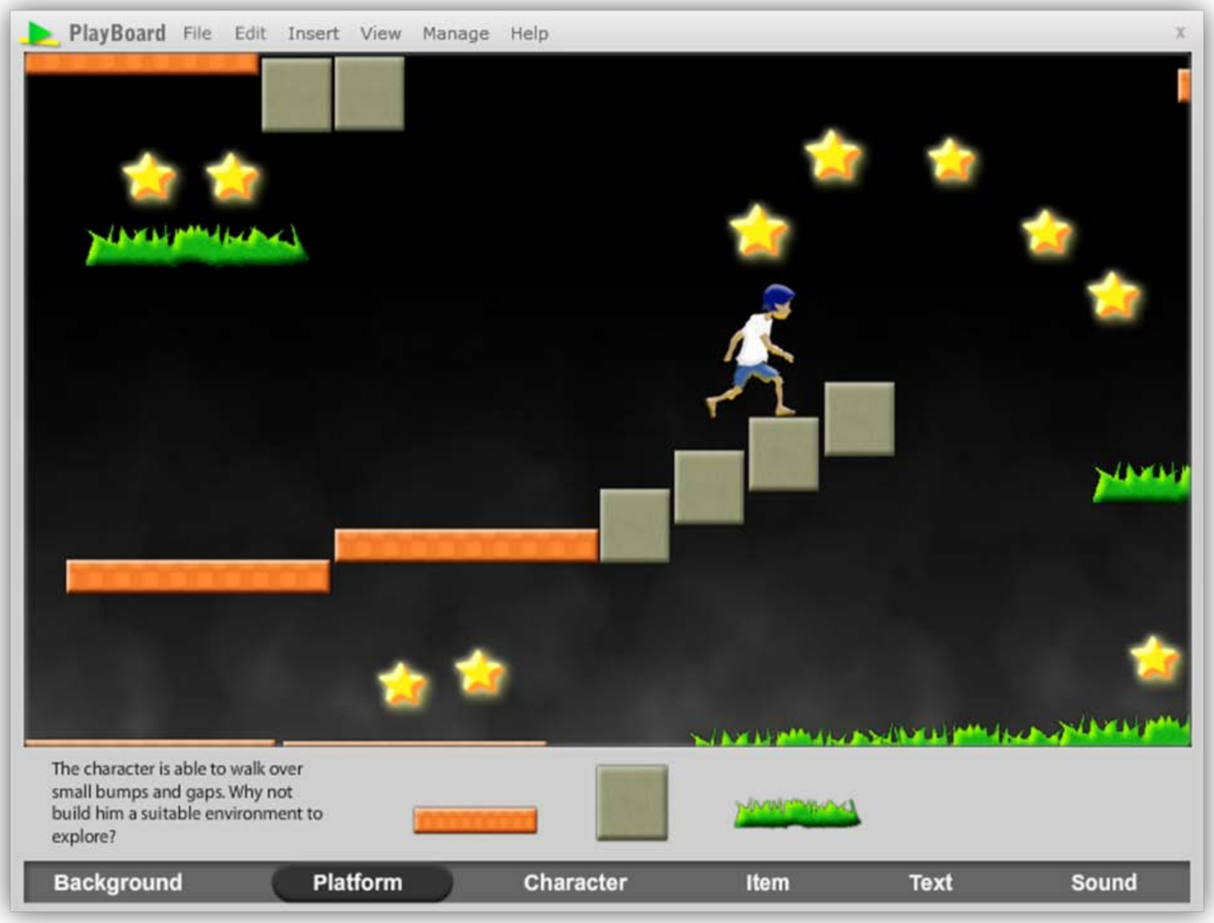
The fourth chapter focused on Goal-Oriented Design, which was compared against User-Centred Design. It demonstrated how Goal-Oriented Design was a desirable methodology because it was unlike User-Centred Design which only focused on inefficient and sometimes unreliable user testing. Goal-Oriented Design is essentially an extended form of User-Centred Design, as it not only includes the user testing process, but also conducts prior research to reduce future misunderstandings. Goal-Oriented Design is also better at understanding its users, being able to make use of fictional personas to effectively simulate the usage and associated effects of its users. Goal-Oriented Design was finally deemed to be more reliable, simply because it was a complete approach with an extensive set of achievable actions, whilst User-Centred Design served best as a checklist than a guideline.

In the design documentation, I described how I made use of Goal-Oriented Design in the creation of a game making tool entitled PlayBoard. This was achieved by following the Goal-Oriented Design's process of Research, Modelling, Requirements and Framework. My research summarised the current conditions of amateur game making as well as highlighted several key areas of difficulty indie game makers faced. In modelling, I created three fictional personas of Colin, Eddie and Sandy who were likely target users of PlayBoard as well as abstracted some of their key concerns in usage. In the requirements section I further made use of these personas to create a usage scenario that attempted to understand the real-time experiences of program usage. Lastly, in the section of frameworks, I accounted the many decisions and actions that have led to the current presentation of PlayBoard.

Ultimately, PlayBoard has been a considerable success. The final usability tests prove that Playboard engages with the target users, albeit to varying degrees. In terms of usability, there was a notable difference compared to common products of game making, however in terms of usefulness, PlayBoard still has plenty of room for potential expansion.

Appendix A. PlayBoard Preview

PlayBoard is a game making tool designed for the non-professional user. It is meant to demonstrate how interaction design principles can affect the way we design complex applications. The program was built by myself within a relatively short timeframe. As a result, this version of PlayBoard is only a prototype of a game making tool with reduced functionalities. However, the remaining functions are results of careful selection and planning, and PlayBoard is nonetheless capable of demonstrating a fair variation of basic games.

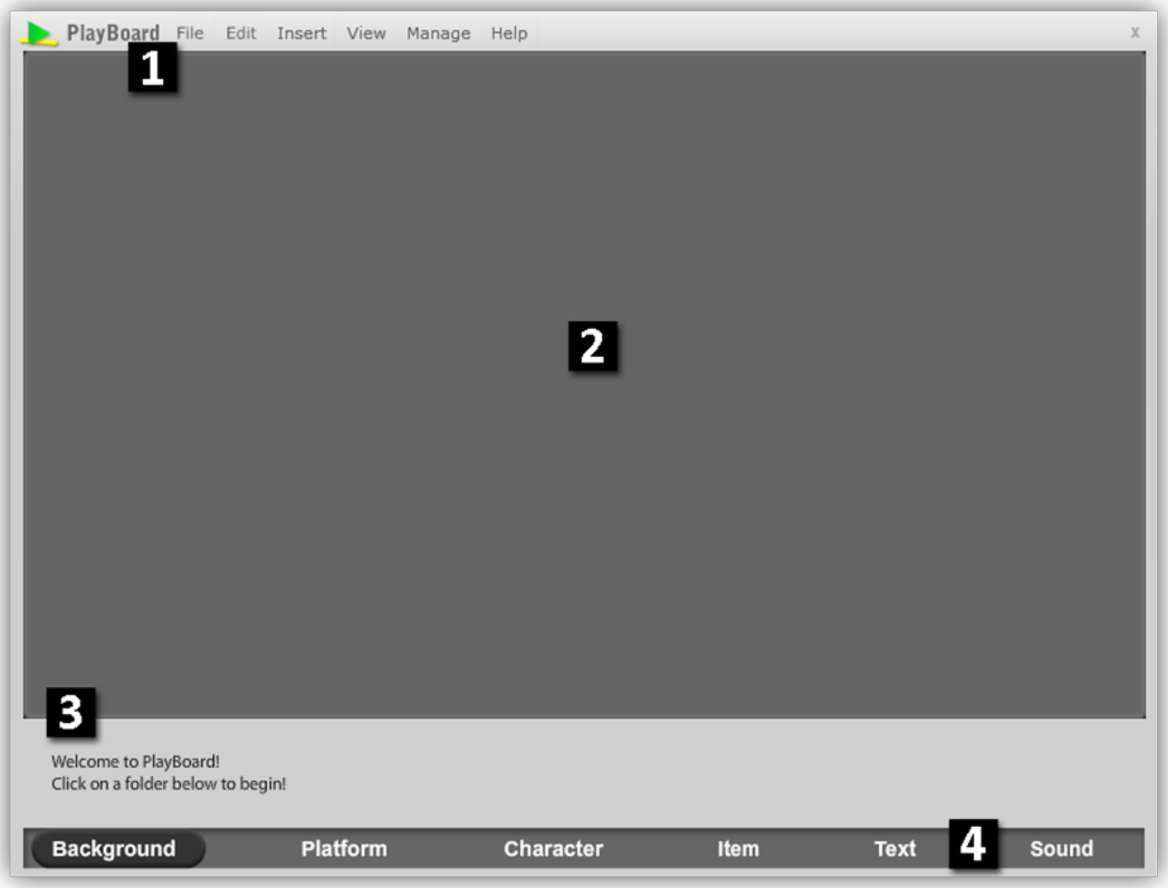


The character is able to walk over small bumps and gaps. Why not build him a suitable environment to explore?

Figure i. Screenshot of PlayBoard.

The PlayBoard Visual Introductory Interface

PlayBoard's visual interface is designed to reflect the progressive activity of usage. Although the starting interface only presents four basic areas, the number of notable interface elements increases depending on interaction.



Welcome to PlayBoard!
Click on a folder below to begin!

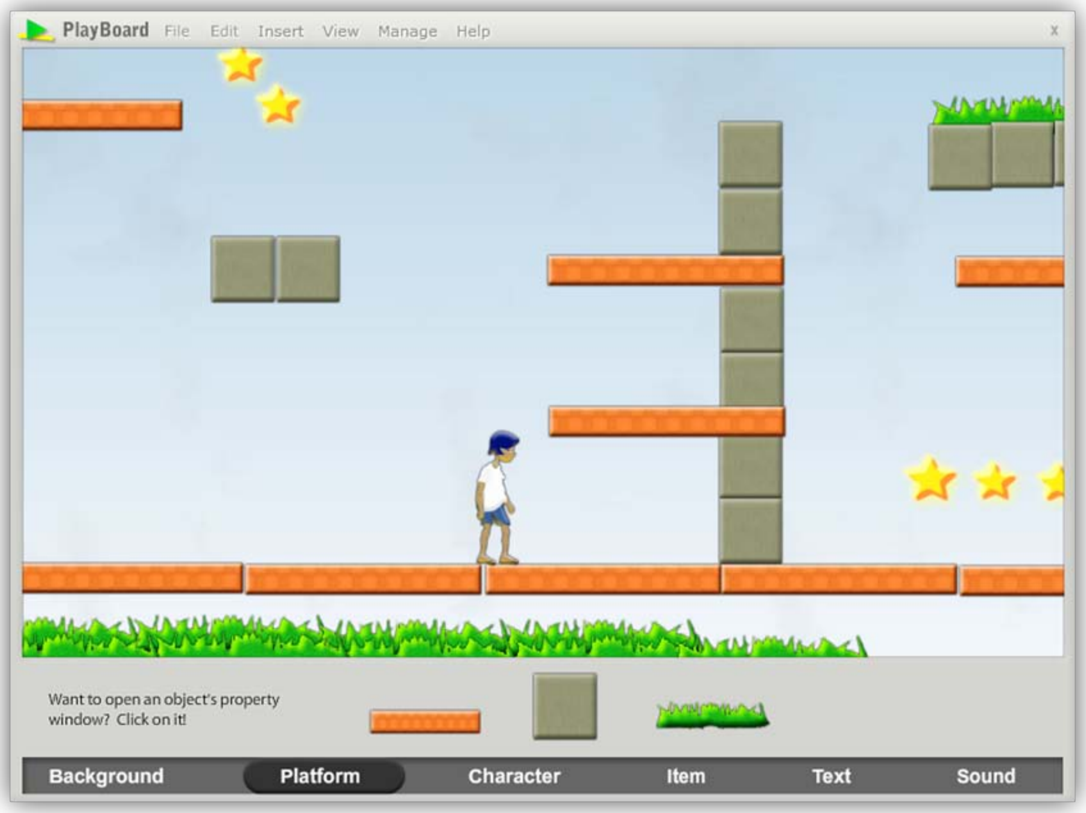
1. Header Bar
This bar contains the PlayBoard logo, example menu buttons, as well as an application closing button. The menus are currently not functional, but are left in the interface to provide a sense of recognition and potential usage.

2. Stage Area
This is the main working area in which objects can be dragged. Objects that are dropped in the stage area are immediately active. Upon selecting active objects, a sub-window of possible functions dynamically appears

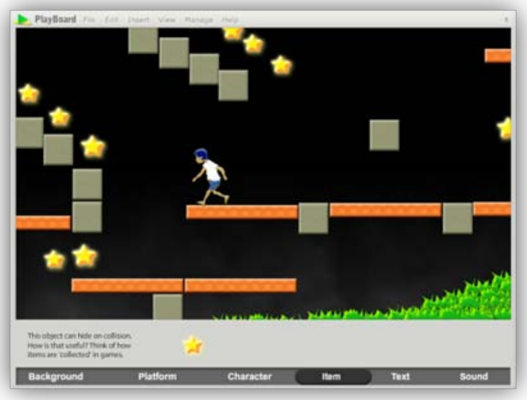
3. Toolbar
This is the tool bar which displays both the prompts and the contained objects within an active folder. An array of possible objects appears when a folder is selected, which can then be dragged into the stage or clicked on for relevant use.

4. Navigational Bar
The navigation bar displays which folder you are currently viewing, as well as the other available areas you can navigate to. Clicking on any one of these icons leads to a presentation of its respective elements.

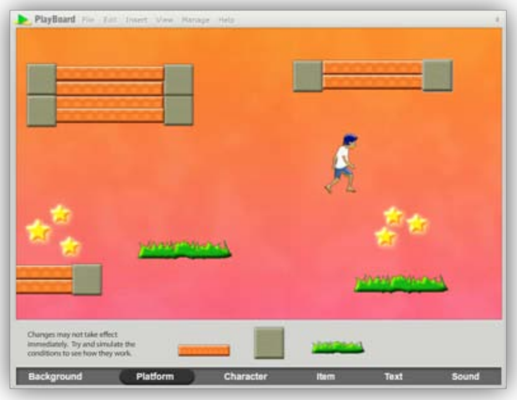
Appendix B. Screenshots from Prototype



Want to open an object's property window? Click on it!

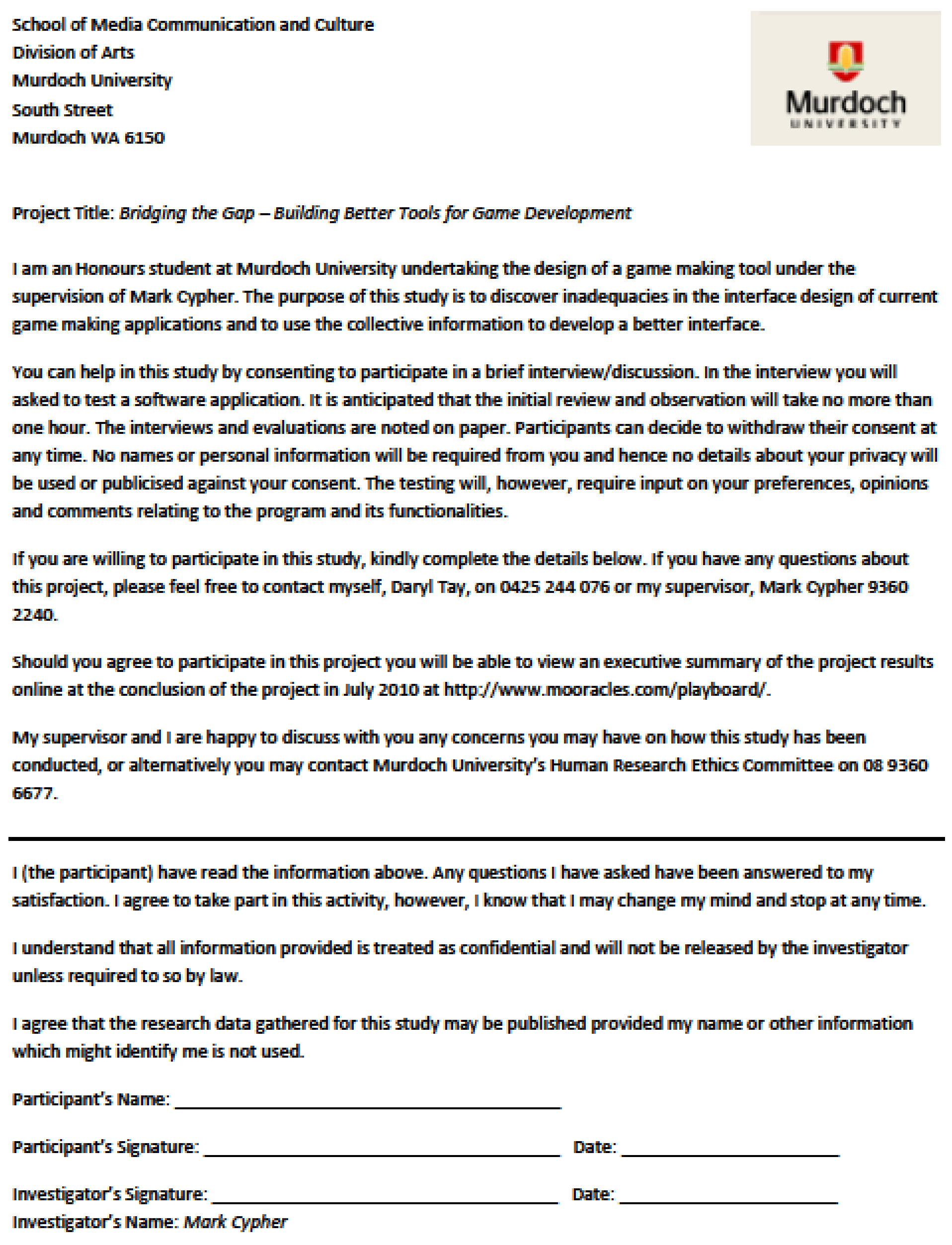


This object can hide on collision. How is that useful? Think of how items are 'collected' in games.



Changes may not take effect immediately. Try and simulate the conditions to see how they work.

Appendix C. Consent Form



School of Media Communication and Culture
Division of Arts
Murdoch University
South Street
Murdoch WA 6150

Murdoch UNIVERSITY

Project Title: *Bridging the Gap – Building Better Tools for Game Development*

I am an Honours student at Murdoch University undertaking the design of a game making tool under the supervision of Mark Cypher. The purpose of this study is to discover inadequacies in the interface design of current game making applications and to use the collective information to develop a better interface.

You can help in this study by consenting to participate in a brief interview/discussion. In the interview you will asked to test a software application. It is anticipated that the initial review and observation will take no more than one hour. The interviews and evaluations are noted on paper. Participants can decide to withdraw their consent at any time. No names or personal information will be required from you and hence no details about your privacy will be used or publicised against your consent. The testing will, however, require input on your preferences, opinions and comments relating to the program and its functionalities.

If you are willing to participate in this study, kindly complete the details below. If you have any questions about this project, please feel free to contact myself, Daryl Tay, on 0425 244 076 or my supervisor, Mark Cypher 9360 2240.

Should you agree to participate in this project you will be able to view an executive summary of the project results online at the conclusion of the project in July 2010 at http://www.mooracles.com/playboard/.

My supervisor and I are happy to discuss with you any concerns you may have on how this study has been conducted, or alternatively you may contact Murdoch University's Human Research Ethics Committee on 08 9360 6677.

I (the participant) have read the information above. Any questions I have asked have been answered to my satisfaction. I agree to take part in this activity, however, I know that I may change my mind and stop at any time.

I understand that all information provided is treated as confidential and will not be released by the investigator unless required to so by law.

I agree that the research data gathered for this study may be published provided my name or other information which might identify me is not used.

Participant's Name: ________________________________

Participant's Signature: ______________________________ Date: __________________

Investigator's Signature: ______________________________ Date: __________________
Investigator's Name: *Mark Cypher*

Appendix D. Questionnaire

Thank you for your cooperation. The following survey is an evaluation for a prototype game making program entitled "Play Board". It is created by an Honours student from Murdoch University under the project "Bridging the Gap – Building Better Tools for Game Development". The answers and comments that you write here will be used as part of a design process to help improve the game making program.

General Information

Kindly mark the most appropriate answer in the brackets provided.

1. Have you ever used or attempted to use **any** computer creative tool such as Adobe Photoshop, Flash CS3, Adobe Illustrator or Game Maker?	a. ( ) Not at all. b. ( ) Attempted, but with little success. c. ( ) Use to a basic degree. d. ( ) Use with moderate skill. e. ( ) Use rather extensively.
2. When using such programs, how often do you encounter usage difficulties?	a. ( ) Not applicable. b. ( ) Frequently, usually unable to solve them. c. ( ) A varying range of difficulties and successes, depending on program. d. ( ) Generally able to find a solution.
3. Are you familiar with game making **programs** or the concept of game making?	a. ( ) Not at all. b. ( ) No, but I would be interested in finding out. c. ( ) Yes, I have some relatable experience. d. ( ) Yes, I have actually been involved in the process of game making.

Program Testing

1. Spend a few moments to familiarise yourself with the interface, click on objects and attempt to figure out the controls. What do you think the program is for?

2. What do you think about the presentation of the interface? Note down any logical or emotional impressions that stand out to you, if any.

3. Attempt to create a game scene with backgrounds and platforms. Build as you see fit, and stop whenever you like. Was this task easy and natural to you? Comment on any difficulties or positive aspects of the design that encouraged/discouraged you from realising your goal.

4. Place a character and a few stars onto the stage. Was this easy to figure out? State whether you succeeded and comment on the difficulty of the task.

5. Attempt to make the character move. Was this easy to figure out? State whether you managed to succeed and comment on the difficulty of the task.

6. Attempt to create the following scenario:
   a) Character moves until a collision with a star.
   b) A jingle sound is played.

   State whether you were successful and comment on the difficulty of the task.

7. Add a few more stars onto the stage and extend the scene accordingly:

   a) Character moves until collision with a star.
   b) A jingle sound is played.
   c) The star disappears.

   State whether you were successful and comment on the difficulty of the task.

8. Using the above environment created, attempt to add the following details:
   a) Character moves until collision with a star.

b) A jingle sound is played.
c) The star disappears.
d) Repeat steps a to d until the last star disappears
e) Display a text "You Win!" on the screen.

State whether you were successful and comment on the difficulty of the task.

9. The tasks given to you so far have acted as prompts to help you design a mini-game. Do you think you would have been able to do so without prompting based on the interface provided? State any aspects of this program that have encouraged or discouraged you from achieving this task.

10. State any other comments, suggestions or complaints you would like to make if you have any.

Appendix E. Citations

Bruns, Axel 2008. *Blogs, Wikipedia, Second Life, and Beyond: From Production to Produsage*. Peter Lang Pub Inc.

Cooper, Alan, Remainn, Robert and Cronin Dave 2007. *About Face 3.0: the Essentials of Interaction Design*. John Wiley & Sons, Inc.

Feller, Joseph, Fitzgerald, Brian, Hissam, Scott A., Lakhani, Karim R, Feller, J. eds. 2005. *Perspectives on Open Source.* Cambridge: MIT Press Books.

Fulp, Tom, and Baez, John 2005. "*Indie power! Riding the FBI with Alien Hominid*". *Game Developer* 12.5 (2005): 28+. Academic OneFile. Web. 19 June 2010.

GRABstats.com 2009. *Online Gaming Statistics, Industry Figures and Information*. Retrieved 23 June 2010 from http://www.grabstats.com/statcategorymain.asp?StatCatID=14

Helgason, David 2009. *A free Unity?* Retrieved 23 June 2010 from http://blogs.unity3d.com/2009/10/29/a-free-unity/

Newgrounds 2010a*. Fun Flash Portal Statistics.* Retrieved 15 June 2010 from http://www.newgrounds.com/portal/stats

Newgrounds 2010b. *The Most Recent Portal Submissions.* Retrieved 15 June 2010 from http://www.newgrounds.com/portal/list.php?which=mr&order=date

Norman, Donald A. 2005. Human-centered design considered harmful. *interactions* 12, 4 (Jul. 2005), 14-19. DOI= http://doi.acm.org/10.1145/1070960.1070976

Norman, Donald A. 1988. *The Design of Everyday Things.* New York: Currency and Doubleday.

Paterno, Fabio. 1999. *Model-Based Design and Evaluation of Interactive Applications*. London: Springer-Verlag London Limited

Sheffield, Brandon 2009. "Download Revolution". *Game Developer* 16.9 (2009): 21. Academic OneFile. Web. 19 June 2010.

SourceForge 2010. *Dev Status.* Retrieved 23 June 2010 from http://sourceforge.net/softwaremap/?words=&sort=latest_file_date&sortdir=desc&offset=75&type_of_search=soft&fq%5B%5D=trove%3A794

Tunnell, Jeff 2009. *Pushbutton for beginners – a possibility?* Retrieved 23 June 2010 from http://pushbuttonengine.com/forum/viewtopic.php?f=5&t=91