Bond University
Research Repository

Procedural literacy:...educating art and design students with code

Birt, James R.; De Byl, Penny Baillie

Published: 01/11/2012

Document Version:
Publisher's PDF, also known as Version of record

Link to publication in Bond University research repository.

Recommended citation(APA):
Birt, J. R., & De Byl, P. B. (2012). *Procedural literacy:...educating art and design students with code*. Abstract from CODE - A Media, Games & Art Conference, Melbourne, Australia. http://code2012.wdfiles.com/local--files/program/CODE%20Abstracts.pdf

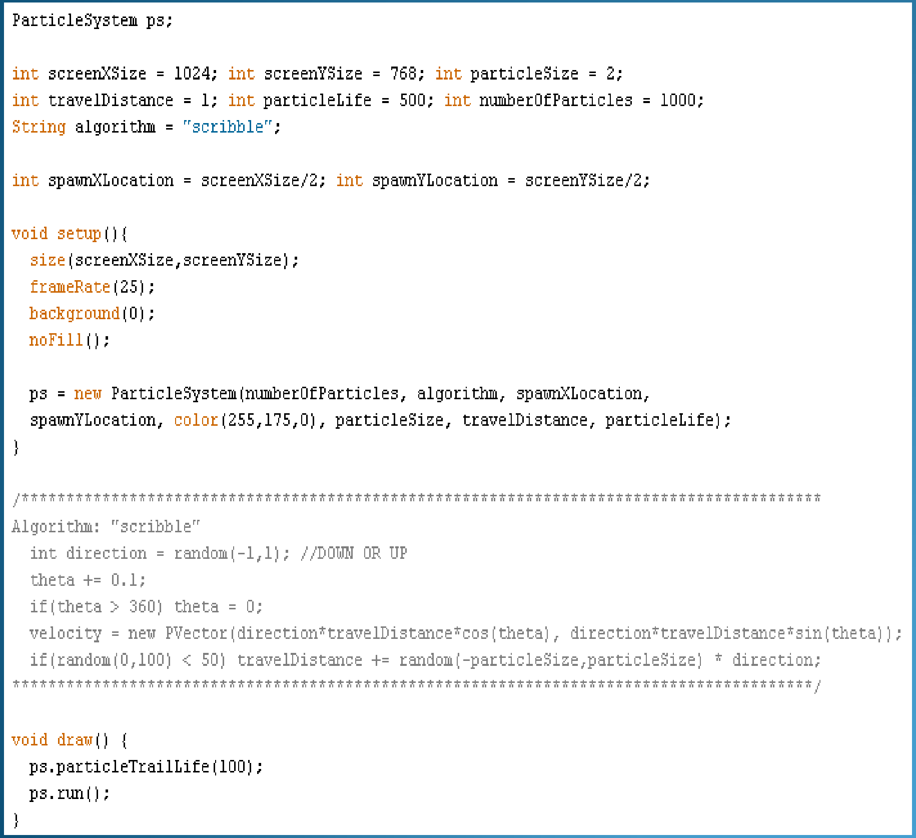
General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

For more information, or if you believe that this document breaches copyright, please contact the Bond University research repository coordinator.

Download date: 09 Oct 2020

Procedural Literacy

How and Why Teachers are Educating Art and Design Students with Code



```
ParticleSystem ps;

int screenXSize = 1024; int screenYSize = 768; int particleSize = 2;
int travelDistance = 1; int particleLife = 500; int numberOfParticles = 1000;
String algorithm = "scribble";

int spawnXLocation = screenXSize/2; int spawnYLocation = screenYSize/2;

void setup(){
  size(screenXSize,screenYSize);
  frameRate(25);
  background(0);
  noFill();

  ps = new ParticleSystem(numberOfParticles, algorithm, spawnXLocation,
  spawnYLocation, color(255,175,0), particleSize, travelDistance, particleLife);
}

/*******************************************************************************************
Algorithm: "scribble"
  int direction = random(-1,1); //DOWN OR UP
  theta += 0.1;
  if(theta > 360) theta = 0;
  velocity = new PVector(direction*travelDistance*cos(theta), direction*travelDistance*sin(theta));
  if(random(0,100) < 50) travelDistance += random(-particleSize,particleSize) * direction;
*******************************************************************************************/

void draw() {
  ps.particleTrailLife(100);
  ps.run();
}
```

Dr Penny de Byl (Associate Professor) & Dr James Birt (Assistant Professor)

Multimedia and Computer Games, Faculty of Humanities and Social Sciences, Bond University, Gold Coast

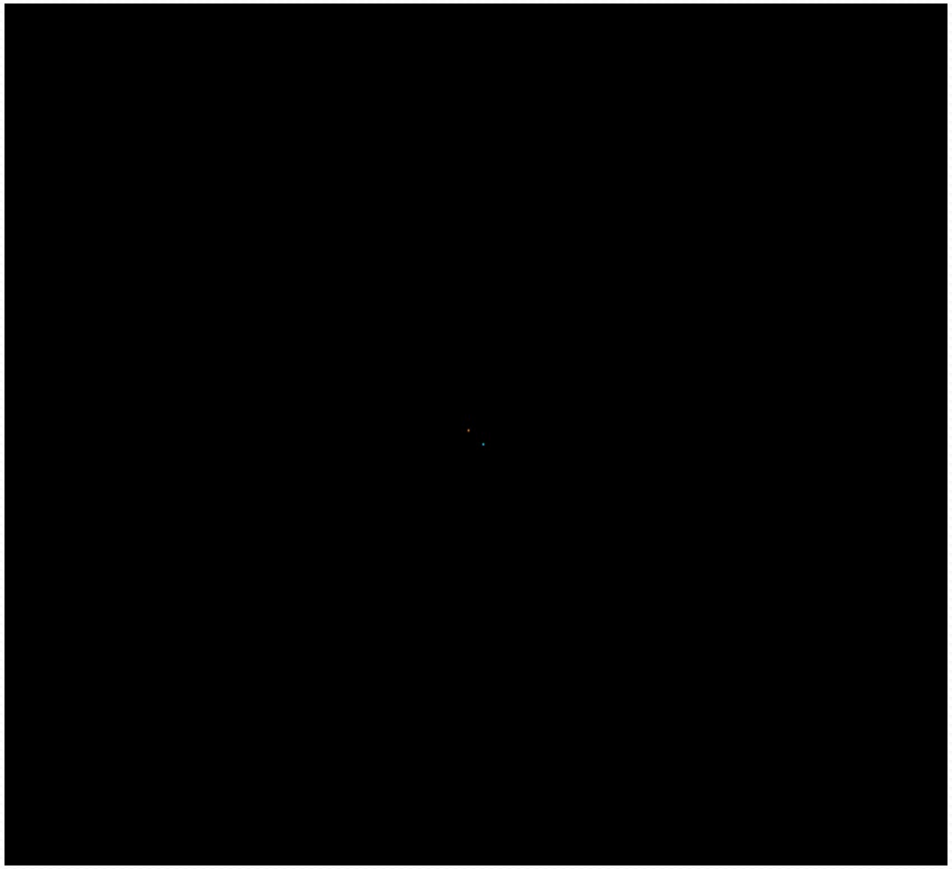
Introduction – About Me

- James Birt – BIT (Hons), PhD
- Assistant Professor Games & Multimedia Bond University
- 10 years teaching experience (Multimedia, Games & IT)
- 5 years industry experience (SE, PM & QA)
- Research Interests (Technology in Education, Serious Games, HCI & Data Visualisation)

Teaching philosophy: Ensure students are innovative, creative, critical and analytical in the pursuit of their chosen profession

What is Procedural Literacy?

The ability to converse and engage through code, rules and design to understand the interplay between culturally-embedded practices of human meaning-making and technically-mediated processes.



```
//Particles start in screen centre then move to a random X,Y using +- travel Distance
if (algorithm == "jitter") {
location.x = location.x+random(-travelDisance,travelDisance);
location.y = location.y+random(-travelDisance,travelDisance); }
```

- Mateas M. 2008. Procedural literacy: educating the new media practitioner. In: Drew D, editor. Beyond Fun: ETC Press. p 67-83.
- Bogost, I. 2005. Procedural literacy: Problem Solving with Programming, Systems and Play. In the Journal of Media Literacy 52,

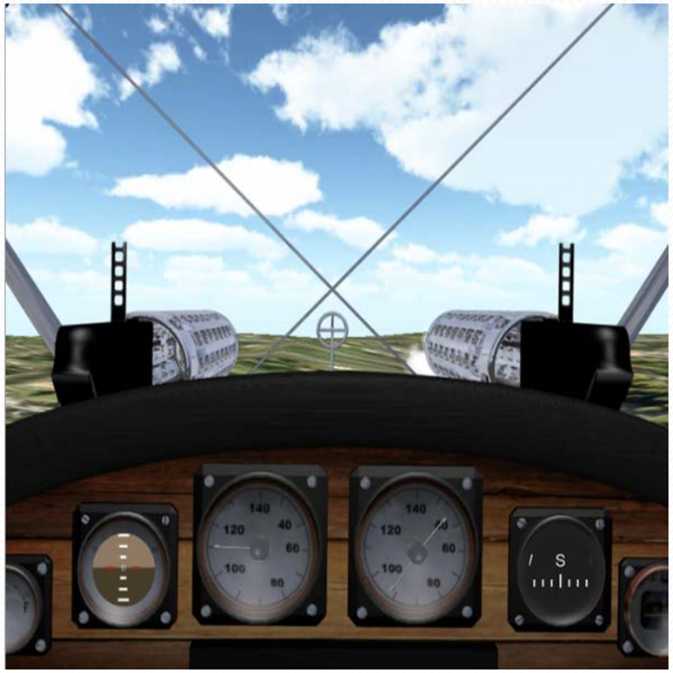
Introduction

Computer Games Student
Bond University (Profile Art)

- A&D students → code is irrelevant, complex & nerdy
- Industry → competitive, multi skilled requiring procedural literacy
- Many papers about individual attempts at teaching code to artists
- No A&D perspective examination of how these attempts are carried out

Background – Student Learning

- Learners incline towards particular modes
  - Arts (Rourke & O'Connor, 2009)
  - Computing (Cagiltay, 2008)
- Challenge → Aligning curriculum, pedagogy, student outcomes & industry standards
- Experiential Learning Theory (Kolb, 1984)
  - Arts: diverging, concrete experience & reflective observation
  - Computing: converging, abstract conceptualisation & active experimentation

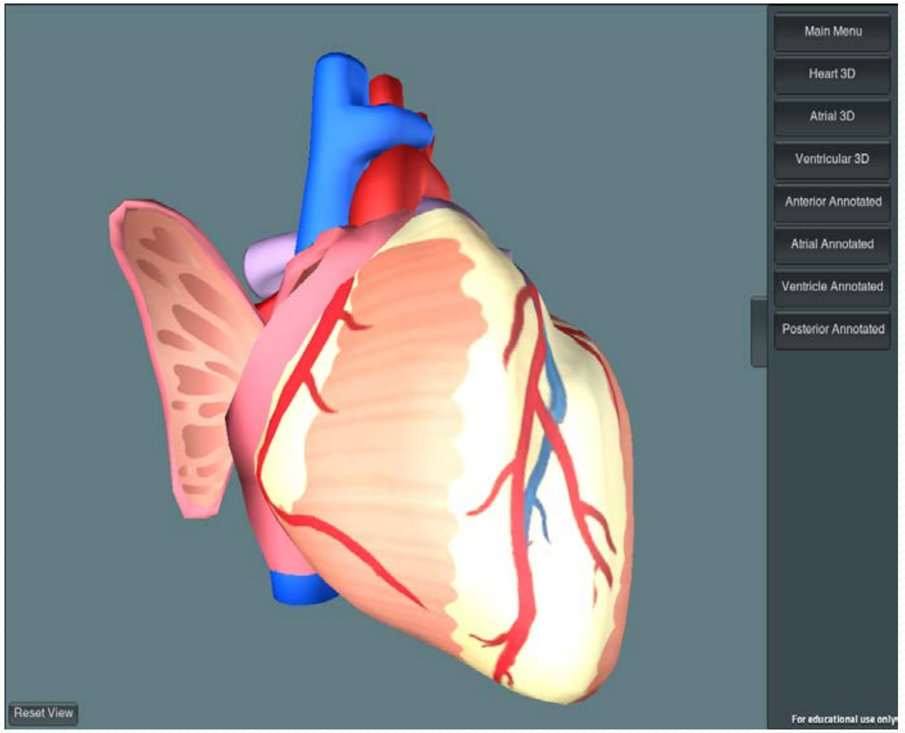


Computer Games Student
Bond University (First 3D Game)

- Rourke, A., & O'Connor, Z. (2009). Look Before You Leap: Testing Some Assumptions on Visual Literacy and Predominant Learning Modalities of Undergraduate Design Students in Australia and New Zealand. International Journal Of Learning, 16(8), 33-45.
- Cagiltay, N. (2008). Using learning styles theory in engineering education. *European Journal Of Engineering Education*, 33(4), 415-424. doi:10.1080/03043790802253541.

Background – Research Studies

- Many studies relating CS & Teaching Code
- Lower number related to A&D & Teaching Code
- Study incorporation criteria
  - Prior Knowledge
  - Why programming was taught
  - Pedagogical Approach
  - Code Tool used
  - Barriers encountered



Computer Games Student
Bond University (RA - Anatomical enriched eBook)

Research Aim → Meta-analysis

- Collate & analyse A&D research experiences
- In-depth surveys from educators teaching code to A&D Students

<u>Study</u>

- How & Why educators are teaching programming to A&D students in tertiary institutions

Computer Games Student
Bond University (First Year Graduated – eLearning Program)

Methodology

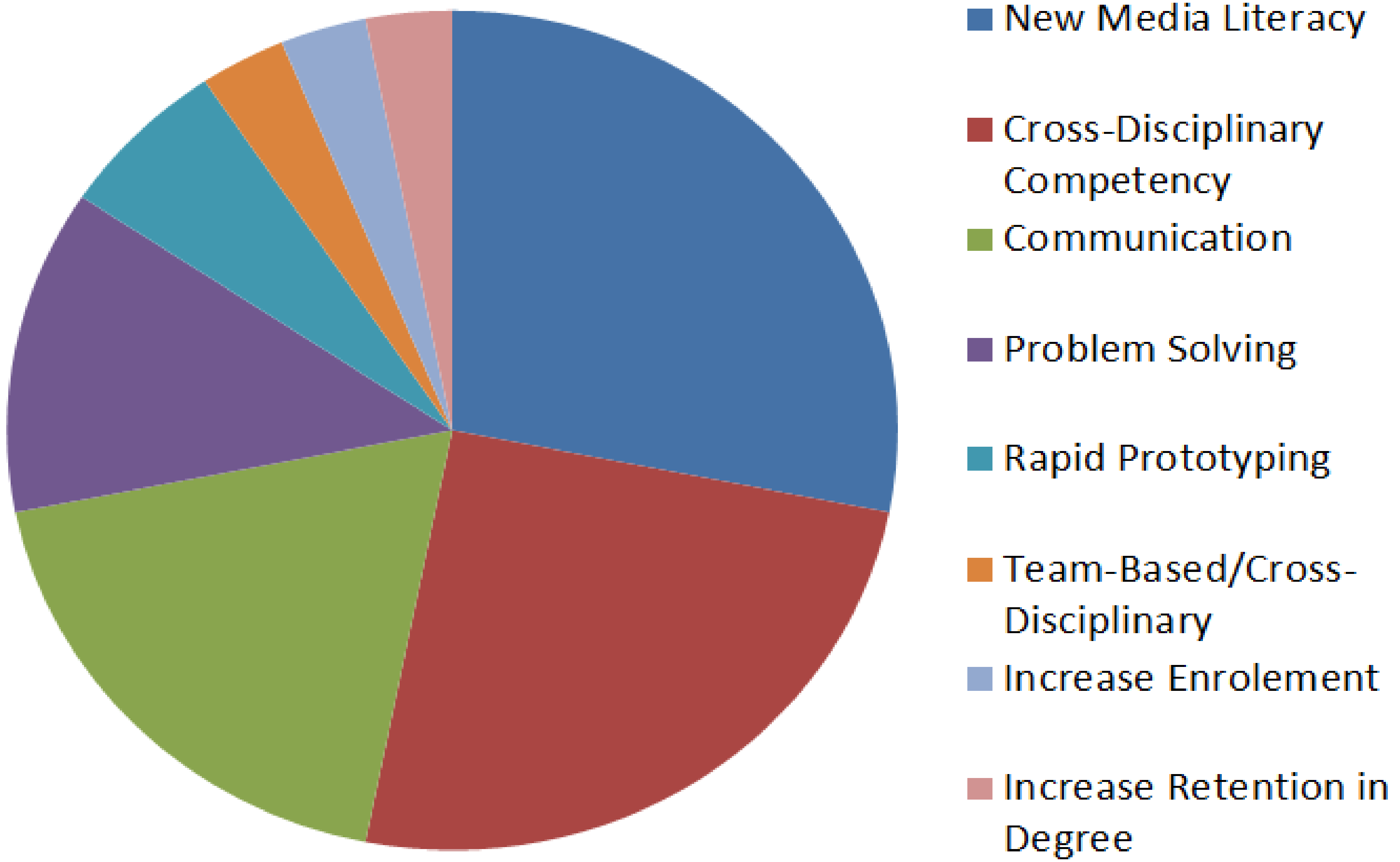
- Jan → March 2012, interviewed 11 tertiary educators teaching code to A&D students
- Worldwide survey recruited via GAMENETWORK & IGDA Games in Education mail groups
- 11 questions → relate experiences of teaching code to A&D students
  - Prior knowledge, Why code taught, Pedagogical approach, Strategies & tools, Barriers encountered, Overcoming Barriers, Effect of code on A&D & Recommendations
- Meta-analysis of survey results with relevant literature studies

Results → Assumed Prior Knowledge

- Entry Level Courses → No prior programming experience
- Advanced Courses → Required entry-level courses

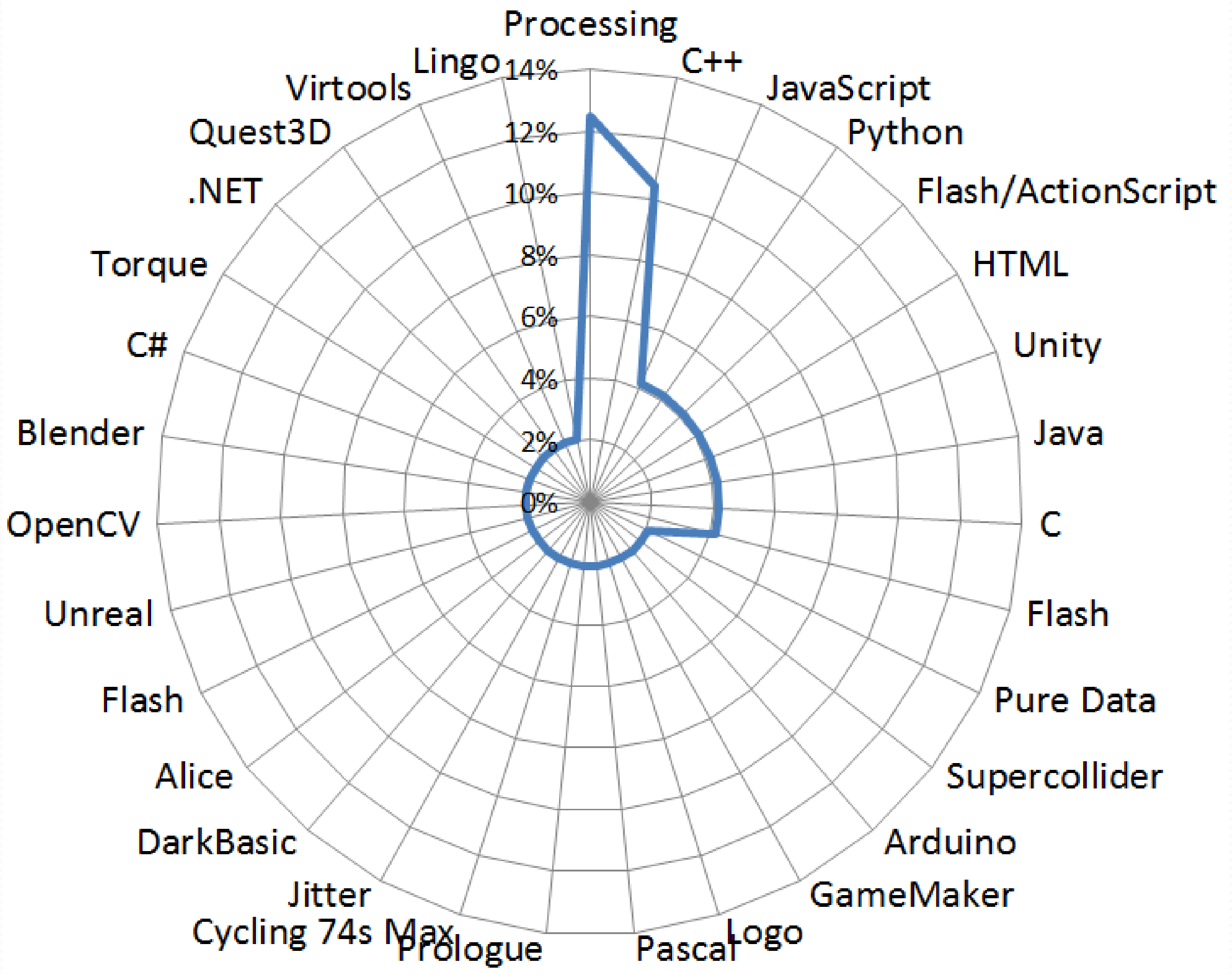
Respondents	Research Study
Four reported students with prior experience	Results inline with research studies. Majority indicated a mix of disciplines & experience levels in the courses
One reported significant class wide experience of 33%	

Why Programming Was Taught



Code Tools Used

→ C++ most used language in gaming (McGill 2011)



- McGill, Monica M. 2011. Motivations and informing frameworks of game degree programs in the United Kingdom and the United States. In Proceedings of the 2011 conference on Information technology education SIGITE 11, 67. ACM Press.

Top 5 Barriers & Strategies

Barriers	Removing Barriers
1. Students Effort	1. Practice
2. Fear → Misconceptions	2. Student Buy-in
3. Frustration → Progress	3. Rapid Prototyping
4. Textbooks	4. Supervised Exercises
5. Programming background	5. Brief or No Lectures

Best Pedagogical Approach

- Visual
- Gently Gently
- Examples/On Demand
- Procedurally/Bottom Up
- Progression through Multiple Tools

Conclusions

- A&D students have initial code fears → frustration & lack of effort
- Overcome → Pedagogical approach: visual techniques, on-demand lectures & student buy-in
- Wide variety of code used → Top 3: Processing, C++, Javascript
- Why Procedural Literacy → multi-skilled new media workplace

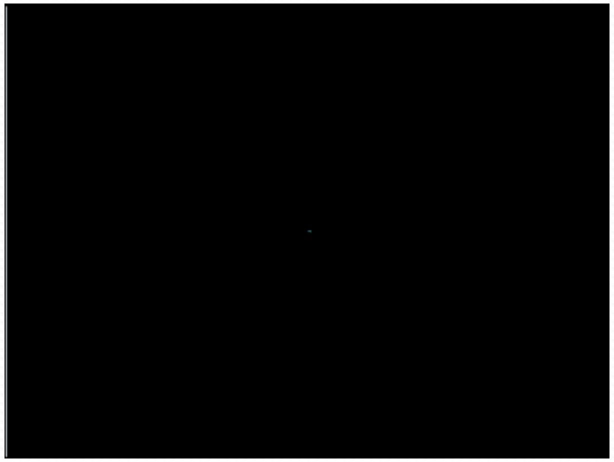
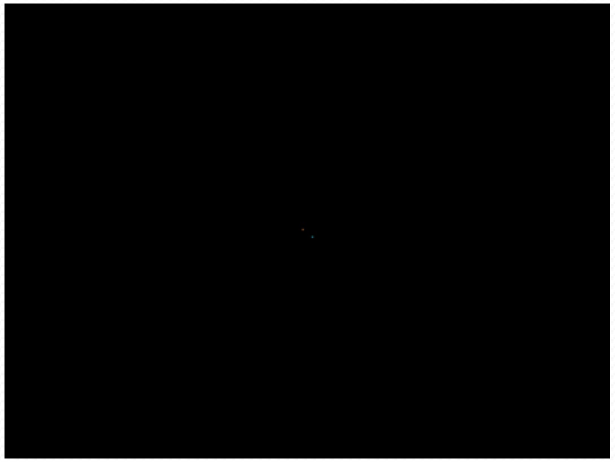
Future Work

- Through this study & restructuring of degree offerings at Bond University a new course Procedural Literacy will become a foundation subject in the new Bachelor of Interactive Media & Design
- Code of Choice: http://processing.org

- Question:
- Can we engage creative design students to improve their motivation for learning to code by using Procedural Literacy?
- Can we improve creative game design students understanding of code concepts by using Procedural Literacy?

Reas, C., & Fry, B. (2007). Processing: A Programming Handbook for Visual Designers and Artists.

Questions?

Research Study References

- Anderson, E. F. and McLoughlin, L., 2007. Critters in the Classroom: A 3D Computer-Game-Like Tool for Teaching Programming to Computer Animation Students. In: SIGGRAPH '07: ACM SIGGRAPH 2007 Educators Program, 05-09 Aug 2007, San Diego, USA.
- Andersen, G, Bennedsen J, Brandorff S, Caspersen ME, Mosegaard J. 2003. Teaching programming to liberal arts students: a narrative media approach. SIGCSE Bull 35:109-113.
- Colvin, John (2005) Teaching Software Development to Non-Software Engineering Students. In: World Congress in Applied Computing (FECS 2005), June 20-23, 2005, Las Vegas, Nevada.
- Heines, J., & Schedlbauer, M. (2007). Teaching Object-Oriented Concepts Through GUI Programming. 11th Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts, held at the 21st European Conf. on Object-Oriented Programming. Berlin, Germany.
- Law KMY, Lee VCS, Yu YT. 2010. Learning motivation in e-learning facilitated computer programming courses. Computers and Education 55:218-228.
- Mateas M. 2008. Procedural literacy: educating the new media practitioner. In: Drew D, editor. Beyond Fun: ETC Press. p 67-83.
- Orr G. 2009. Computational thinking through programming and algorithmic art. In: SIGGRAPH 2009: Talks. New Orleans, Louisiana: ACM. p 1-1.
- Tsai M-H, Huang C-H, Zeng J-Y. 2006. Game programming courses for non programmers. In: Proceedings of the 2006 international conference on Game research and development. Perth, Australia: Murdoch University. p 219-223.
- Wills, GB, Davis, HC and Cooke, EC (2004) Paired Programming for Non-Computing Students. In, LTSN-ICS Fifth Annual Conference, Ulster, UK

Interview Questions

1. Did you assume any student prior knowledge of programming before they began the subject? If so, what?
2. Why was programming taught to these students? How do you feel about teaching programming to design students? What is the benefit of such classes?
3. What strategies (e.g. pedagogical approach) and tools (e.g. game engines, programming languages) were used? Please list and align with different student cohorts where applicable.
4. What strategies and tools worked well for you? Please elaborate.
5. What would you do differently next time? Please explain why.
6. What strategies and/or tools should be discontinued? Why?
7. What were some barriers, if any, that you encountered? Student engagement? Lack of technical assistance?
8. How did you over come these barrier(s)?
9. What effect if any, do you feel programming concepts have had on design students? Did you notice attitude changes in the cohort towards the more technical side of computing? Did any students produce unexpected outstanding work? Did any students go on to do more programming?
10. What strategies and tools would you recommend to other teachers having to teach programming to design students in the future?