Lowering Development Barriers in Educational Game Design

Austin Bart, Robert Deaton, Eric McGinnis
Virginia Tech, University of Delaware

Introduction

Creating a game is an excellent learning opportunity for computer science students because it requires an in-depth understanding of programming languages, object-oriented design, problem-solving, constraints, and teamwork. Games are also a powerful teaching tool which primary and secondary school educators have used for targeted teaching, improvement, and evaluation of their students' skills. In response, the University of Delaware created a course called Educational Game Development which works with middle school teachers at the Chester Community Charter School (CCCS) [Burns et al. (2012)]. Teams of 3-5 upperclassmen are given the opportunity to apply skills learned in the classroom to create software for real-world clients. To help this course achieve its goals, we have developed a set of tools to address some of the difficulties and limitations in existing tools.

Development Platform

Five years ago, CCCS received 1,400 XO laptops, low-cost computers developed by the One Laptop Per Child Foundation which run a custom version of Linux and are programmed in Python. The Python package Pygame is used for game development. Because Python and Pygame are cross-platform, students are able to develop, test, and deploy their games on Windows, OS X, Linux, and XO computers.

Spyral

Spyral is a 2D sprite-based engine built on top of Pygame designed to allow the rapid development of games, particularly those targeted at low performance platforms. Spyral started as a library to simply optimize drawing because of the limited performance of the XO, and grew to include many features that had to previously be written by nearly every team. Spyral now includes modules to handle:

- Images - loading and drawing graphics
- Sprites - positioning images on the screen
- Cameras - batch sprite manipulation
- Scenes - game organization
- Fonts - loading and rendering text
- Events - keyboard and mouse input
- Vectors - manipulating 2D coordinates
- Animations - simple sprite transformations
- Clock - controlling game speed
- Forms - higher level input handling

In addition to providing tools for making game development easier, Spyral also works to teach and encourage better software engineering practices to the users, targetting places where poor decisions were routinely made in the past.

Spyral (continued)

In particular, the use of a scene system encourages the separation of content into disjoint pieces where possible, and the event system is designed to mimick and be used like modern event driven systems for games, networking software, and parallel software.

Conspyre

Conspyre is a cloud-based networking system built to work in conjunction with Spyral. The system is designed with two parts: (1) a client library for XO games that talks to (2) a web framework that can store and retrieve data and provides a portal for teachers. Using this sytem, XO developers can persist data between students' play sessions and enable communication between teachers and students. As it is used primarily by novice developers with limited experience, Conspyre is written in Python and built on a scaffolding paradigm by which students can quickly develop functional applications with a minimal knowledge of web development.

Example.activity

Example.activity is a template for organizing games written using Spyral and other libraries for easy deployment to the OLPC XO as well as testing on a user's regular computer. The core features are

- A launcher made specifically for running on the OLPC XO
- Bundled libraries like Spyral, conspyre, sugargame, and their dependencies
- Generating and bundling translations
- A launcher made specifically for development, which includes
  - Options for resolution changing
  - Profiling code to find performance issues
  - Opening a debugger on crashes
- Uploading of stack traces to a conspyre server to debug issues for end users

Platipy

The development of applications with rich graphical user interfaces has traditionally required an in-depth knowledge of a platform's programming languages and frameworks. Platipy is a documentation project which seeks to leverage domain knowledge common to most Junior and Senior-level computer science students to teach Python and Spyral in a fun, quick, and interactive way that allows users to begin making games as soon as possible. Platipy covers basic concepts such as obtaining software, setting up a development and testing environment, an introduction to Python (including syntax, data structures, functions, and classes) and Spyral documentation. The introduction to Spyral culminates in an example game which includes important components such as adding and controlling graphics, creating game logic, detecting collisions, and handling user input. These elements form the basis for developing both simple and complex games.

Teacher Surveys

In order to measure whether there was improvement as our resources were integrated into Educational Game Development, we selected a subset of the games from semesters of class before and after the introduction of Spyral and Platipy. A group of teachers from CCCS then examined the games and rated their reactions in four different categories: how fun are the games, how well do the games evaluate a student's skills, how well does the game teach students new skills, and how useful the game would be in their classroom. The average of the survey responses are shown in Figure 1, divided by whether Spyral was used in the creation of these games. Additionally, we asked the teachers to choose the two games which they believed to be the best, and the results are shown in Table 1.

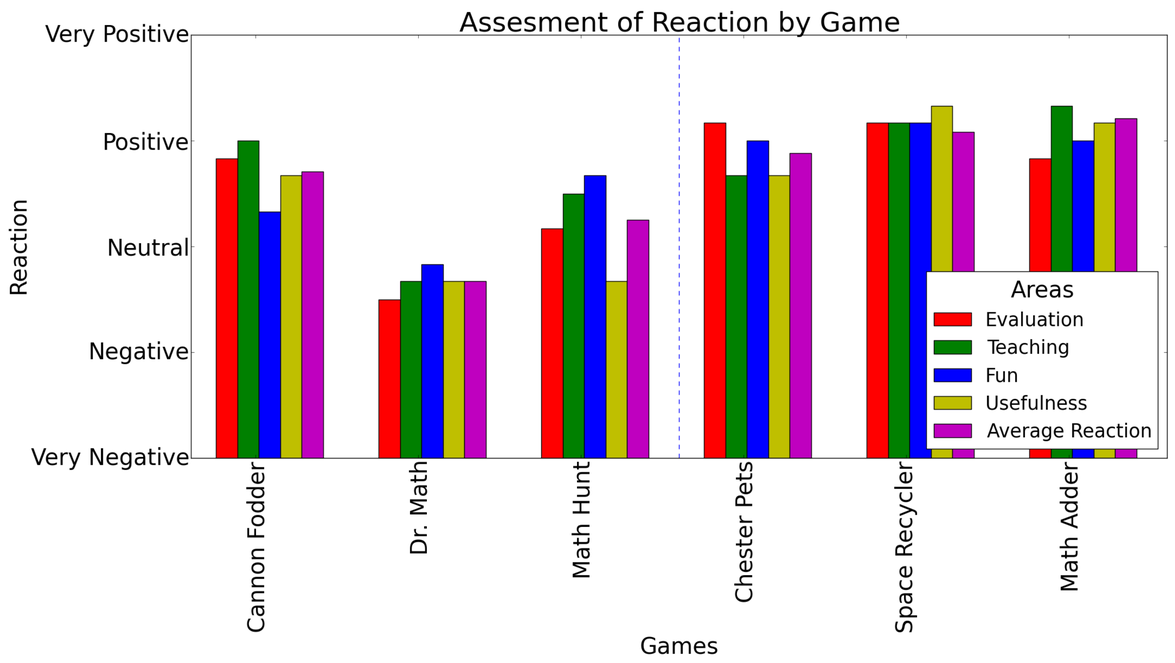

Figure: Results of a survey on a group of CCCS teachers on reactions to games developed. Games to the left of the dotted line were written in semesters before Spyral was used in Educational Game Development.

Results

Game	Most Fun	Best Teaching	Most Useful
Dr. Math	0	0	1
Math Hunt	1	2	1
Cannon Fodder	0	1	1
Space Recycler	2	4	3
Math Adder	4	1	2
Chester Pets	5	3	3

Table: Games chosen as top in each category from a survey of 6 teachers.

Conclusion

The goal of the Educational Game Design class is for students to gain hard programming skills, soft human interaction skills, and experience the value of writing software with a real-world impact. In the past, programming has required a majority portion of the course and reduced the amount of time that can be spent eliciting input from teachers and designing, testing, and deploying games. The creation of Platipy, Spyral, and Conspyre have reduced the amount of time undergraduates spent learning frameworks and optimizing their games, allowing them to focus more on their soft skills and the high-level educational game design process. As evidenced by the data gathered from CCCS teachers, the quality of games created after introducing our tools has increased significantly, leading to students with a better appreciation for the impact of educational games.

Future Work

There is significant room for improvement in Platipy, Spyral, and Conspyre going forward. We plan to expand Platipy to include more complete examples, including downloads of projects from previous semesters, and better coverage of the full Spyral API. We also plan on compiling a list of useful resources for game programmers including information on developing basic AI, incorporating multiplayer, and obtaining royalty-free artwork. Spyral will have additional features added based on the feedback from each semester of the course. Lastly, Conspyre will include more complete examples and the potential for an offline mode that allows syncing when network access is available.

References

Burns, Richard, Lori Pollock, and Terry Harvey (2012), "Integrating hard and soft skills: software engineers serving middle school teachers." In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, 209–214, ACM, New York, NY, USA, URL `http://doi.acm.org/10.1145/2157136.2157199`.