Developing Educational Software for the XO-1

Nolan Baker and Dr. Mark Goadrich (advisor)
Mathematics and Computer Science Department
Centenary College of Louisiana

Introduction:
One Laptop Per Child, Inc. was founded in January 2005 with the goal of creating educational opportunities for the poorest children in the world by providing them a ruged, low-cost, low-power laptop. This laptop, the XO-1, runs Fedora Linux with a revolutionary new GUI interface called Sugar, designed specifically for children, so when developing software for the Sugar environment, one thing must be kept in mind; connect and educate.

Methodology:
Sugar is written Python, so, naturally, that's the language we chose for writing our activities. From there, our choice of graphics library was pretty obvious. The Pygame library provides high-level access to low-level tasks like using the keyboard and drawing on the screen. It's simple, robust, well documented, and already had a strong code base in the Sugar API. With each program, we sought to dig deeper into the capabilities of Sugar.

Conclusions:
All of these programs have reached a few thousand downloads so far, and we can only expect more as they mature beyond beta stages in development. The Sugar community is consistantly posting bugs to be fixed. All of the code is open source, so anyone who wants to help out with the code can. Research is far from over, though. We'll continue to explore the capabilities of the XO-1 with 3D graphics, artificial intelligence, and simplified interfaces. Code will become faster and more efficient. And translations are still to come.

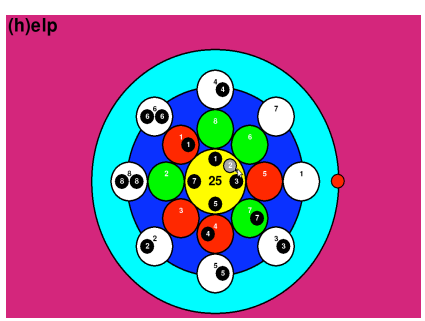
Special thanks to the Centenary College Student/Faculty Summer Research Program, sponsored by the Office of the Provost for funding and support. We couldn't have done it without you.

References:
http://wiki.laptop.org/
http://www.sugarlabs.org/
http://www.pygame.org
http://wiki.laptop.org/go/Cell_Management
http://wiki.laptop.org/go/Space_Tag
http://wiki.laptop.org/go/COBBLE

Cell Management:
This was our first attempt at writing a game using the Pygame library. Cell Management, originally a Piece-Pack game by Dr. Mark Goadrich, is a logic puzzle based on the regulatory mechanisms present in DNA. The object of the game is to help one of each species escape to the center. The rules translated fairly easily to code:

```
while the number of escapees < the number of species:
    for each cell starting with the one clicked on:
        if (the adj. hiding space is red and empty or
            green and occupied):
            if corresponding hiding space is empty:
                prisoner moves to corresponding hiding space
            else:
                prisoner escapes
        move guard clockwise
        if prisoners are in the adjacent hiding space:
            return them to their cell
```

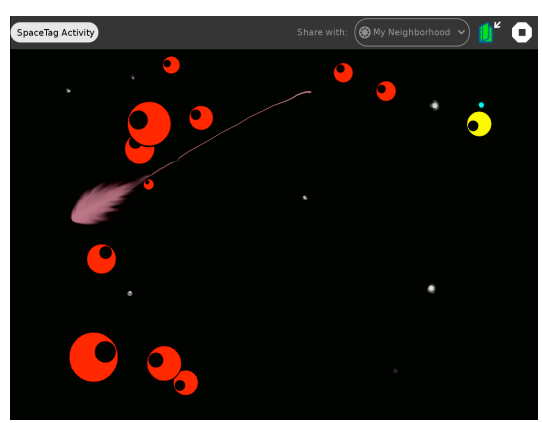
The original game had 6 cells, but it was clear from the beginning that the learning curve was much too steep to start off with this many cells, especially since it was to be played by 6-year-olds without instructions. The solution to this was to start off with a simple 2-cell game and increase the number of cells as each puzzle was completed. This did cause a few problems when we found out that some of the games were unsolvable, but all of these problems were resolved.



SpaceTag:
Once we had sucessfully completed a game and had a better working knowledge of the Sugar API and Pygame, it was time to move on to sharing, one of Sugar's primary features. The design of SpaceTag is simple, play tag in 3-space with realistic physics. You play as a yellow ship viewed from above, your friends are red ships. The ship that's 'it' is indicated by a blue dot that circles it.

Rather than have one player be the server and have everyone else connect to the server, we wanted the game to be as modular as possible, so even if the player who starts the game loses his or her connection, the game continues for everyone else. Our solution to this was Telepathy, an instant messaging system. All events in the game are passed as messages which are interpreted by each player. When a player joins the game, they are essentially starting their own game and telling everyone else where they are. When everyone else gets that message, they create a dummy that updates its position acording to the message.

Currently, I'm working on deploying AI controlled bots that take advantage of flocking and pathfinding algorithms to give you the feeling of playing with your friends without actually requiring you to have friends. Hopefully this will be finished by the time this poster is being read.



COBBLE:
With two games under our belt and a firm grip on networking, our next task was COBBLE (COllaBrative Boardgame Learning Environment), a finite mathematics toolkit slash super fun play pit. The idea was to provide all of the pieces to make a game and leave the rules to the child. COBBLE would then give mathematical feedback on what's happening in the game. Our goal was to teach the basics of finite mathematics through the use of simple toys like cards and dice.

Currently implemented are dice, cards, boards, tokens, and bags, all of which can be customized to fit your game's needs, and shared with your friends. Also implemented is a chat box for communication and feedback. All of these were built from scratch with a custom graphical user interface. Of course, this led to some problems with text input which turned out to be quite a cumbersome task, but this was relatively minor compared to the problems with networking.

Unlike the networking for SpaceTag where only one type of object was created (the dummy), COBBLE required a much more robust system. With COBBLE, each object could be created and manipulated by anyone playing the game in real time. Python's interpretive capabilities provided a simple, elegant solution to our problems. Python's eval() function allows for reading strings as code on the fly, so creating objects that appeared on other people's screens was as simple as broadcasting a string that looked like code.