**Title: Parabolic Game Suite**

**Category: PyGame Game DELUXE SUITE**

**Repository:** [**https://feef@bitbucket.org/feef/parabolic**](https://feef@bitbucket.org/feef/parabolic)

**Summary: A suite of several games, including GALAXE, an old fashioned space invaders game, Defender, a game in which the player flies through a canyon that continually gets smaller while shooting at aliens and collecting powerups, Pong, which is kind of obvious, Tanks, which is a multiplayer game in which two tanks drive around and shoot at each other, and more as the project progresses, will all be included on one disc. All games will be accessible through an interactive menu system. Most of these games will have a HighScore table that appears on the screen when the game is over. This can be accomplished via pickling and unpickling. All these games will provide for a variety of fun activities for users.**

**Also on the disc will be a Python 3.1.2 32-bit download for Windows, a PyGame download for Windows, and a LiveWires download for Windows. In the final stage, it should expand to Linux operating systems as well, and possibly even Mac OSX. All of these downloads and all of the games will be automatically installed via an autorun program. The autorun program will be written as a .bat for windows and .sh for linux that will put the game files onto the computer.**

**Implementation: For each game, there will be several classes necessary, most of them based off a games.Sprite class and in some games a *game* class will be made to create all sprites necessary and start the games.screen.mainloop.**

**For each game, everything that moves will be a class. Some games will have a game class to tie everything together. Most games will have a class for the player’s object that checks for keyboard input in the update method. Other classes can be sorted into enemy, environment, weapon, and group categories. Enemies will mostly be immobile objects that fire at the player. Environment classes will be a variety of things, from powerups to walls to teleporters. Most will have a function to kick out anything that is overlapping, like a wall. Weapon classes would be almost identical for all games. They would have a dx or dy so they would only move in one direction. When there is something in weapon.overlapping\_sprites the object will have its die method invoked and it will be destroyed. Then the projectile will be destroyed. Group classes will create a bunch of one class and make them move. Most sprites will have pause methods, so that everything on screen will freeze when the game is paused.**

**HiScore tables will be added by pickling data and unpickling and sorting it out. This can easily be accomplished with lists. An installer will be created as well. For Linux systems, a .sh or .py file will be used to install everything. For Windows users, a .bat file will run the .msi files for python, pygame, and livewires respectively. Then it will proceed to install all the games.**

**The menu system is made up of several menus and sub-menus. There is a base menu that branches off into sub-menus, which branch off again until the desired game is selected. These menus can run other menus or games by implementing the os module. The menu programs are made up of 2 sprites, a select and a menu. The select is primarily a manipulative image that is used to highlight the selected choice. The menu takes user input in its update method to cycle through choices. It creates an instance of select and changes the y value of the select according to what is currently selected. It obtains this information by a series of Boolean variables, one for each selection. When the return key is pressed, a file is run through os.startfile. The file that is run depends on which choice currently returns True.**

**Teamwork: The size of the Parabolic project alone allows a multitude of students to all work on different parts of the project without ever interacting. There are many games which are independent of each other, and many classes in all of the games, which don’t overlap for the most part. If there are too many people working on the games, then there are still the installation features to work on. In event of minor clashes, the students involved and the Project Leader would settle the dispute. In the event of a major clash, everyone on the team would be called for a tribunal deciding what to do on the matter. As team size increases, the number of people on one or more features can be expanded to lessen the work load per person.**

**In the last tier, Chad has been developing games for the most part. Andrew has been debugging games and working on the menu system. In the final stage, Chad will continue to add games to our suite and will work on the installer and menus as well, but not as much. Our project leader, Andrew, will be focusing on helping the rest of the team. He will also be making games and working on the installer. He will work on the menu system alongside Chad. Team members newly recruited will be working on the installer files, graphics, and/or learning pygame, depending on where that person’s talents lie.**