Tony Chun
CSCI 102 | HW#3a
Due Date: 03/28/2011

Design Document

Purpose/Overview

To create a video game using our understanding of C++ and the Qt Creator application/parallel.

Requirements

The five items listed in the homework page on this website (http://www-scf.usc.edu/~csci102/assignments_s12.html) are:

1. Design Document
2. 3 screens/levels
3. Three “special” things: shoot directly at you, follow you
4. Quality of scrolling “things”
5. Player score maintained and displayed on screen
6. Variation in scrolling item movement
7. Player lives managed/game restart on death
8. TA’s enjoyment in playing your game
9. Code Comments

Classes

USC_Student
- the main character that would serve as the vessel for playing the game

USC_Experience
- the main file to hold all the rest of the classes on one platform, and hence building the game

Object
The main parent class to make all the objects necessary for the game other than the student

Grades class has:
1. Letter A Class
    a. This was the letter A that had a 4.0 GPA
    b. Was stationary so that the user had to go get it
2. Letter B Class
    a. This was the letter B that had a 3.0 GPA

        b. Moved vertically
    3. Letter C Class
        a. This was the letter C that had a 2.0 GPA
        b. Moved horizontally
    4. Letter D Class
        a. This was the letter D that had a 1.0 GPA
        b. Moved Diagonally
    5. Letter F Class
        a. This was the letter F that had a 0.0 GPA
        b. Followed the user's cursor
        c. Was the basis for the lives of the player

Specials class has:

1. Cheat Class
    a. 50/50 chance that you would get an A or get caught cheating, which would result in a game over
2. Extra Credit Class
    a. Adds 0.05 to the GPA
3. Freshman Forgiveness Class
    a. Adds an extra life (in this case, it would be an additional F buffer life)
4. Hangover Class
    a. Items speed up until you catch another letter
5. Office Hours Class
    a. Items freeze until you catch another letter
6. Sleep Class
    a. Items slow down until you catch another letter
7. Study Class
    a. Creates another A for you to catch
    b. Arguably the best special in the game

Global Data/Functions

```
void resetState();
```

put the items at a random position on the playing field

```
void autoMove();
```

allowed for the movement of the images by translation

```
void moveUp(int);
```

item moves up

```
void moveDown(int);
```

```
   item moves down

void moveLeft(int);

   item moves to the left

void moveRight(int);

   item moves to the right

void setXDir(int);

   set the horizontal velocity

void setYDir(int);

   set the vertical velocity

virtual int getXDir() = 0;

   This function was inherited by all subclasses

virtual int getYDir() = 0;

   This function was inherited by all subclasses

bool isCaught();

   Checked to see if there was an intersection

void setCaught(bool);

   Set Boolean to true if intersected

void setLetter(int);

   Set GPA or effect necessary

virtual double getLetter() = 0;

   This function was inherited by all subclasses

QRect getRect();

   Returned the rectangle shaped by the program
```

```
    void setRect(QRect);
```

Develop the size of the rectangle

```
    QImage & getImage();
```

Return the image itself.

High-level Architecture

What is most important about the architecture of the polymorphism used in developing this game. There is one root parent class, Object, two subclasses of Object, which are Grades and Specials, and then children are listed above. This is crucially important for later functions, if someone chooses to go into the gaming field, because it allows for the programmer to give “sort of” characteristics to each instanstiated object.

User Interface

The instructions are given at the beginning of the game. Dragging the SC will start the game and the SC will follow the user’s cursor position.

Test Cases

How do you plan on testing your system?

Aside from the obvious “find where the program fails to continue, fix it, and re-run again”, I had a few people play my game.

What are the potential problem cases?

If this is run on parallel, there are bound to be lagging problems that I have no control over.

What are some nominal test cases?

Because there is always room for error, if someone stays in the safe zone long enough, the F will be able to permeate through the safe zone and eventually get to the SC.