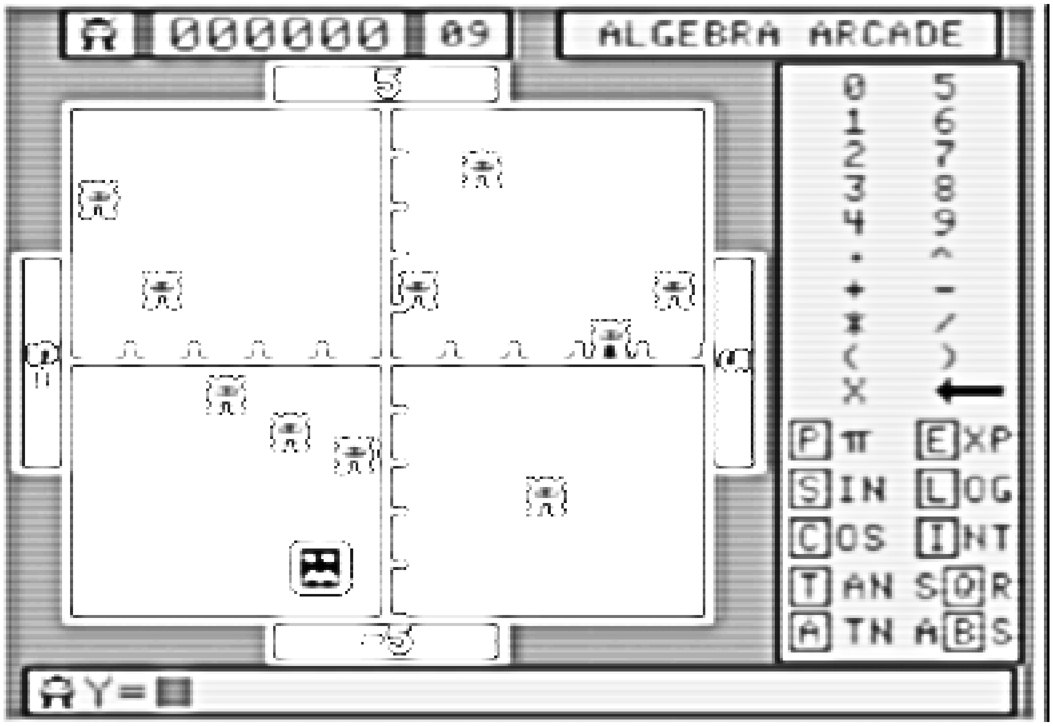
CS 7B - Spring 2018 - Assignment: Adapting the calculator for an Algebra Arcade Style Game using SFML. due 4/27/18



Background Theory

In class, we have developed the tools for writing a game similar to the 1980's era AppleII game, Algebra Arcade.

Broadly speaking the game has the following design specifications:

- A coordinate plane spanning $(-5, 5) \times (-5, 5)$ is displayed in a graphics window with coordinate axes to indicate the x, y-coordinates of points in the plane. There is a set-up feature that allows the players to rescale the plane.
- Ten targets (algebroids) are distributed on the screen with evenly spaced x-coordinates and random y-coordinates. At the same time, one or two players are created with a scores initialized to zero.
- An anti-target, "Graph Gobbler", is displayed at a random location on the screen. This (let's say it's optional) feature completes the set-up for the game.
- The user is presented with an opportunity to input a formula specifying y as a function of x using standard function types like +-*/ and also `sqrt`, `sin`, `cos`, `tan`, `atn`, `abs`, `exp`, `log`, and `int`.
- The game responds to the player's input by plotting the user's function $y(x)$, or what part of it coincides with the field of play.
- After the function is plotted, a "whirlwind" (any kind of animated tracer) retraces the function curve.
- If the whirlwind, while traces the function, hits an algebroid, this event is detected and the algebroid is removed from the playing field.
- Provided the whirlwind doesn't hit the GraphGobbler, the player's score is increased by the formula $200T$ where T is the number of algebroid targets hit in the turn.
- If the Graph Gobbler is hit, an animation showing the the GraphGobbler retracing (and erasing) the graph drawn so far, is shown.

- There is a mechanism keeping track of how many turn have been taken and ending the game if there are no more turns.

Submit code and and text file to answer these questions to my email address.

1. Suppose our boss requires that we take our sketch starter code (as developed in class) and make it conform to the following model:

```
class Game {
public:
            Game();
    void    run();
private:
    void    processEvents();
    void    update();
    void    render();
private:
    sf::RenderWindow mWindow;
    sf::CircleShape mPlayer;
};

int main() {
    Game game;
    game.run();
}
```

   - (40) Submit a reorganization of the code that fits this model.

Starter code is given on the calendar. You don't have to use that, but if you don't, please thoroughly document the authors of the code, which is a standard in any case.

The following features are worth various point-values and may be developed independently of one another in many cases, though there are clear dependencies either required or desired:

- (10) Expand the function options to include `tan`, `arctan`, `int` and `abs` and any other type you like.
- (10) Control the speed of the grapher (to make it go more slowly).
- (10) Distribute 10 algebroid sprites in the playing field.
- (10) Animate the distribution of the 10 algebroid sprites.
- (10) Allow the user to use a backspace to edit their entry.
- (10) Change the playing field to indicate the bounds of x and y coordinates at the extremes of the axes (by default, $-5 \leq x, y \leq 5$
- (10) Reset the playing playing field to get repeated attempts at hitting the algebroids.
- (10) Write a whirlwind sprite to retrace a function after it has been plotted. Don't worry about handling collisions here, but control speed so it doesn't go too fast.
- (10) Detect whether the path of the function collides with an algebroid of not. If it does, remove that algebroid from the playing field and, if there's a score, update the score.
- (10) Place a GraphGobbler sprite at a randome location on the screen.
- (10) Handle the possibility that the whirlwind hits the GraphGoblbler.

- (10) Introduce a second player and allow the players to take turns, each keeping track of their own set of algebroids and their own score.
- (x10) Other significant improvement features, as approved.