Ermi Girmai & Satvik Saini
Section F

Snake Encounter

Problem Description: Our program will be recreating the Snake Game. Users will play with a snake and will be required to capture/eat the snack in order to keep playing. However in pursuit of the prize the user's snake will grow in length. The user's score is dependent on the amount of snacks it captures. But wait, when do you lose?! If the user's snack runs into itself then the game is over. Have fun!

Data Description:

```
// The Player class is where the players position, their direction, and their length.
class Snake : public Drawable{

        private:
                int x; //This is to track the position of the snake's head
                int y;// Snake's tail (which grows)
                int z; // Random number generator for the food
                int score; // To keep track of the score
                char map; //full size map of the game
                bool L, U, D, R; //Controls for the game L=Left, U= Up, D=down, R=right
                int speed;
                Bar &b;

        public:
                Snake();
                Snake(Bar &b, int x, int y, int speed); // This is the parameterized constructor
                which will have all the private data member functions as parameters.
                int GameOver();
                void SetDirection(int direction);
                void draw(RenderTarget& rt, RenderStates rs) const;
                void Move(Map &map);
                int directionIAmGoing;
                int directionIWannaGo;
};

// The Direction class sets the possible directions to numbers so that the code can call each
//direction by its name
class Direction{

public:
        static const int NORTH = 0;
        static const int EAST = 1;
        static const int SOUTH = 2;
        static const int WEST = 3;
};

// The Bar class is located at the top of the window and will hold the number of points the player
//has collected
class Bar : public Drawable{

        private:
                int playerPoints;
                Font font;

        public:
                Bar(Font font);
                void draw(RenderTarget& rt, RenderStates rs) const;
                void AddPoints(int playerPoints);
                int GetPoints() const;
};

// The Map class draws the window where the playing field will be mapped.
class Map : public Drawable{

        public:
                static const int ROWS 10;
                static const int COLS =10;

                Squares GetSquares(int, int) const;
                void SetSquareFood(int z);
                void LoadFile(string);
                bool NoMoreFood();

        private:
                Squares square[ROWS][COLS];
};
// The Squares class draws each square of the playing field and each squares attribute
class Squares : public Drawable{

        public:
                static const int SQUARE_WIDTH = 50;
                static const int SQAURE_HEIGHT = 50;

                void SetPosition(int x, int y);
                void SetColor(Color color);
                void SetSize(int w, int h);
                void SetWalkable(bool w);
                bool IsWalkable() const;
                int GetPoints() const;
                void draw(RenderTarget&, RenderStates) const;

        private:
                int x, y;
                Color color;
                int width, height;
                bool walkable;
                int food;
};
```

Procedural Description:

```
Create Window
Load graphics and recall the controls for snake
Initialize variables
        create a 2D array representing the playing field
        call head from class
        call tail from class
        create food from class
Main Loop
        if left, up, down, right arrow key pressed
                Set head of the snake according to the key pressed
                Move the snake according to direction of the snake's head
                Move the snake's tail to follow the snake's head
        if snake's head position == food position then
                increment score
                increment the snake's length
                randomize the location of the next food position
        if snake's not dead for certain amount of time
                double the score and set it equal to the bonus

        draw food //using SFML
        draw snake head and tail according to their position
        illustrate the current score in the top right corner

        if snake's head runs into the body of the snake
                print game over
                print the score to a data file //data file to track high scores and etc

end main loop
```