b

School of Sciences and Engineering

Fall 2018

CSCE110- Frogger Project

Initial Design and Documentation

**Group name:** Int Elligence

**Team Members:**

Mohammed Abuelwafa ID #900172603

Shrouk Beshara ID #900171136

Marwa Sulaiman ID #900172284

**General Requirements:** “**All of these are well explained in classes and pseudo code”**

1. **Menu** to start/exit the game, change options (sound on/off), user login and high scores (the maximum 5 high scores in the game overall)
2. **The game:**

* A frog controlled by the keyboard WASD and the arrows.
* 3 levels differ in the difficulty according to the time limit, speed of the cars, logs, turtles and the frog, pickups available and the design of the Game (The number of roads as an example)
* **The player has 3 lives, the frog dies in one of the following cases:**
* The frog hits a car
* The frog falls in the water
* The frog goes to one end of the screen on a log or a turtle
* The time limit ends.

and re-spawns again from the starting position at the bottom of the screen if it still   
 has another life or finish the game of no lives left.

* the floor, water, roads, cars and theme may change according to the level.
* The game shows “**Game Over**” screen for 5 seconds, then go back to the menu. If the player wins, the game should **show the score** for 5 seconds, save it in the high scores list and go back to the menu.
* The score is going to be calculated as following:

1. Add 50 points for each finished level.
2. Add the remaining time to the score.
3. Add extra 10 for each remaining life.
4. Add 30 for each taken coin.

**Proposed Bonuses:**

* **Pickups:**

1. Coin: +30 in the score
2. Heart: add one life
3. Drink: dizzy frog (slower movement)
4. Clock: adds extra 30 seconds.

* **Gameplay:**

Multi player or single player options may be provided.

Multiplayer: either **2 frogs in the same screen** or **two separate windows** or **One window divided into two halves** (right and left for each player).

* **Networking:**

Online gaming experience will be enhanced using the networks library (SFML/Network.hpp).

* **Options:**

1. the ability to turn on or off the sound during the game.
2. the ability to restart while playing.
3. Shop to buy extra pickups unlocked by specific scores.

**The Algorithm of the Game:**

* A main menu with a list of the buttons (start, options, High scores and Exit) appears as the user opens the game. If the user presses on options a window with the following buttons appear:

1. Sound ON
2. Sound OFF

* If the user presses on start, the game starts from level 1 and the grid is drawn as 2D array from the loaded file (a file for each level) by iterating over the characters in the folder (0 represents floor, 1 represents water, 3 represents road). A rectangle of size 20x 20 is drawn for each sector.
* The window of the game will contain a text and a number for the user’s score and lives and a timer for the remaining time (The timer may not be applied in all levels)
* The frog will be placed in a determined place in the 2D array (First row in the bottom half of the screen in the middle)
* The frog would be a rectangle shape with a PNG texture of size of 20\*20 and it would move a step of 20 in any direction (x++, x-- , y++y--) (A block in the 2D grid in each move).
* Each row of cars would be 1D array of cars that moves with determined spacing distance and speed according to a clock for that row (The clock and speed (i.e. speed) may differ in more difficult levels)
* Each row of logs and turtles would be a 1D array that moves with determined spacing and speed according to a clock for each row (The clock and speed (i.e. speed) may differ in more difficult levels)
* The rectangles that represent the start and end of a road or the start and end of a row in a river (i.e. tunnels) would be represented with a special texture in which when a car, log, turtle reaches or intersects half of it appears and the half disappears and then it vanishes totally**. The tunnels are restricted areas for the frog that he cannot move in.**
* The frog is known to die when it intersects a car or a tunnel or fall in the water (using function intersect in the class of Rectangle shape) and then lives decreases by 1. If there is no left lives ( a message of game over appears), but it there is extra lives it respawns in the bottom of the screen and the player is known to finish the level when it intersects the upper row of floor in the top screen and then the next level is loaded from the next file using the 2D array grid.
* The score is incremented calculated after the end of each level using the previously mentioned criteria.

**Classes and structure:**

the game would have a main.cpp file in which all the variables, objects, streams, enums and structs are created from the corresponding classes or libraries and all the objects and variables are passed to the game class ( to be processed using the functions of the game class ). when a variable or object is needed to be passed, it would be passed to the functions of the class that needs it ( for example passing the score to messages ).

1. **Game:** this class has all the functions that it is associated with creating the game and the levels. For example:

* Initialize
* Handle events
* Game Loop

Pseudocode:

Class Game

{

public:

void initializeLevel1 (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text, sound[]);

void initializeLevel2 (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text, sound[]);

void initializeLevel3 (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text, sound[]);

void Level1 (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text, sounds[] );

void Level2 (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text, sounds[] );

void Level3 (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text, sounds[] );

void handleGameEvents ( frog & Player, renderwindow & window);

void drawGame ( (renderWindow &window, rectangleShape [][18],frog&player, obstacle car[] , obstacle turtle[], obstacle logs [], int& lives , int& score, clock& cars, clock &turtles , clock &logs, clock& pickups, pickup[],Text& text , sounds[] );

}

1. **Pickups:** this class has all the functions that is associated with the pickups for example

* Create coin
* create heart
* create clock
* create drink

Pseudocode:

Class pickup

{

private:

rectangleShape pickShape;

texture pickpic ;

public:

pickup ();

void drawPick ( RectangleShape ); //in which the texture is set and the rectangleShape of the pickup is drawn

rectangleShape GetShape () const;

void setPick ( rectangle Shape , int PickX , int pickY);

~pickup  
}

1. **Obstacles:**  this class has all the functions associated with creating the obstacles for example:

* Create car
* Create log
* Create turtle
* Create row of cars
* Create row of turtles
* Create row of logs

Pseudocode:

Class Obstacle

{

private:

rectangleshape ObsShape;

texture pic;

public:

obstacle();

void drawObs ( rectangleShape);

void moveObs ( rectangleShape, int stepx, int stepy);

rectangleShape GetShape ( ) const;

~obstacle();

}

1. **Messages:** this class has all the messages that are going to appear to the user in different situations for examples:

* Print\_Win
* Print\_GameOver
* Print\_HighScore
* Print\_Exit
* Print\_timeleft
* Print\_lives

Pseudocode:

Class Messages

{

private:

text Msg;

font MsgFont;

public:

void printGameOver ( &RenderWindow );

void printWin (&RenderWindow);

void printExit ( &RenderWindow ) ;

void printScore (&RenderWindow ) ;

}

1. **Audios:** the audios are going to be played by declaring and loading a buffer in the main and creating a variable ( sound track1 ) then ( track1.setbuffer ( “mp3 file” )) and then the functions of ( play , stop, pause ) are going to be used.
2. **Main Menu:** this class has all the different buttons and events associated with the main menu. For examples of functions:

* StartGame
* Open\_options
* Open\_HighScores
* Open\_Exit

Pseudocode:

Class MainMenu

{

Public:

MainMenu ();

Void createMenu(renderWindow&window);

Void drawButton (&rectangleshape,renderwindow& menu, int x , int y );

Void setTitle(&Text,&Font);

Void handleEvents( renderWindow&, options[])

~MainMenu();

Private :

renderWindow window;

Rectangleshape options [4];

Texture optionspictures [4];

Text title ;

Font font ;

}

1. **Frog:** this class has the frog and all its associated function as:

* setTexture
* Move
* Die
* Dizzy

Pseudocode:

Class Frog

{

Private:

RectangleShape frogShape;

int score;

int lives;

Texture frog\_pic;

Texture Hearts\_pic ;

public:

Frog();

void setFrogTexture( Texture&frogtexture);

void setposition( int x, int y);

RectangleShape getShape () const;

void drawFrog( RectangleShape);

void setlives( int lives);

int getlives() const ;

void moveFrog ( RectangleShape )

int setscore(int score, bool (// if pickups are eaten ), time, remaining lives));

bool isdead(); // true if it is in water or intersects with the cars or goes to one end of

the screen

bool intersect\_Car();

bool intersect\_Obs ();

bool intersection\_Pick ();

~Frog();

};