Initial Project Design Documentation

Names:

Eman Darwish ID# 900172070

Zahraa Gamal ID# 900171683

Nour Ibrahim ID# 900161744

General Requirements:

● Menu to start/exit the game, change options (sound on/off), user login and high scores.

● The game:

- A player controlled by the keyboard.

- 4 levels.

- The player have 3 lives, dies after it hits any of the ghosts and re-spawns again if it still has another life or finish the game of no lives left.

- the tiles, maze, theme changes according to the level.

● Two types of Pac-Dots, the small dot for collecting points and the other for 5 seconds immunity. In addition, there will be pop-up dots that boosts the score.

● 4 consecutive levels with different parsing text files and themes.

● Ending screen with either “Game won” with the amount of coins collected if passed all the levels or “Game Over” if lost all the lives before (either by time or moving ghosts) that goes back to the menu afterwards.

● The score will be always appeared on screen.

Algorithms:

-A main menu with a list of options (Start, controls, and exit) appears as the user opens the game.

If controls is pressed, a window with:

Q: to quit game.

P: pause game.

M: mute the music.

If Exit is pressed then close the window.

-If the user presses on start, the game starts from level 1 and the grid is drawn as a 2D from the loaded file by iterating over the characters in the file and for each # a rectangle of size 20x20 and left empty with just colour filled the number of cols and rows depends on the loaded map.

-The window of the game will contain a text and number for the user’s score and lives. Also, the window will display the pacman in an initial position determined in the array along with the ghosts(each one will move randomly in the array) and the two types of dots (sprite with texture) will be distributed in the places where there exists no rectangles. -The positions of the big dots will be distributed randomly in the array and will have max number of 3. The number (4) and position of the ghosts will be set. If the number of dots remaining is less than the total number of dots in the grid and the lives is less than 3, the game should continue. When the user presses the keyboard arrow keys the position of the pacman in the array will be incremented or decremented ( x--, y--). If the pacman intersects a small dot, increment the score by 20 and dots eaten disappear.If big dot the same applies in addition to changing the shape of the ghosts and if the pacman intersects the ghosts within 5 seconds from intersecting the pacman is not dead( bool false).If the pac man hits a ghost, the lives will decrease and then check if lives is less than 3 ( the pacman not dead) if true then go back to the first condition to the same level and current score.If false, then print the text (you lose). If the dots= total dots in the map and lives< 3, this means that the level is up and move to the second level.

If level is = 2, then change the map loaded for the grid to be harder containing more rectangles(barriers) and the texture of the rectangles to change the theme of the level same applies for

for levels 3 and 4. Also, the ghosts will move faster. Repeat the game again for the different map.

Classes:

Class # 1 : Pac-Man (The Player)

Class # 2 : Dots

Class # 3 : Game Levels

Class # 4 : Main Menu

Class # 5 : Ghosts

Class # 6 : Pop-Ups

Class # 1 : Pac-Man (The Player)

{

Private:

rectangleshape pac;

long score;

int lives;

texture pic;

Public:

PacMan (); // construction

Void getposition();

Void setposition(vector2f);

Void draw (rectangleshape);

Void moveUP();

Void moveDown();

Void moveRight();

Void moveLeft();

int setlives();

int getlives():

int getscore ();

Int incrementscore( score);

bool isdead ();

}

Class # 2 : Dots

{

Private:

Rectangleshape dot;

Texture pic;

Bool eaten;

Public:

Void draw (Rectangleshape, int) ;

Bool isEaten();

}

Class # 3 : Game Levels

{

Private:

int number;

PacMan p;

Dots d[20];

Ghosts g[5];

Rectangleshape grid [20][20];

Texture score ;

Texture lives;

Public:

Level ();

Void gameloop (renderwindow&);

Void setlevel (int number);

int setlevel ();

Void getlevel ();

Void drawscore (score);

Void drawlives (lives);

}

Class # 4 : Main Menu

{

Private :

Rectangleshape options [5];

Texture optionpic [5];

Text title ;

Public:

MainMenu (); // constructor

Void drawoptions (rectangleshape []);

Void setTitle();

Void getTitle;

}

Class # 5 : Ghosts

{

Private :

Rectangleshape ghost;

Texture pic;

Public :

Ghosts (); // constructor

bool isimmune( );

Void drawghost (rectangleshape);

Void moveghosts (rectangleshape);

}

Class # 6 : Pop-Ups

{

Private:

Texture pic;

Rectangleshape pop;

enum type= {1,2 };

Public:

Pop-Ups();

Void drawPops (rectangleshape );

Void setPosition( );

}