Version 5.6

January 19, 2019



Created By: Abinash Singh

# Balloons Tower Defense

## eclipse

### Setting up ide

For Eclipse it is much simpler to include the needed libraries

### Required Libraries)

| Libraries | name | link |
| --- | --- | --- |
| Lwjgl\_util.jar | Lwjgl util | Same as main and is located inside the zip file |
| Slick – util.jar library only | Slick texturing | http://slick.ninjacave.com/slick-util/ |
| Lwgl.jar | Lwjgl main | https://sourceforge.net/projects/java-game-lib/files/Official%20Releases/LWJGL%202.9.3/lwjgl-2.9.3.zip/download |

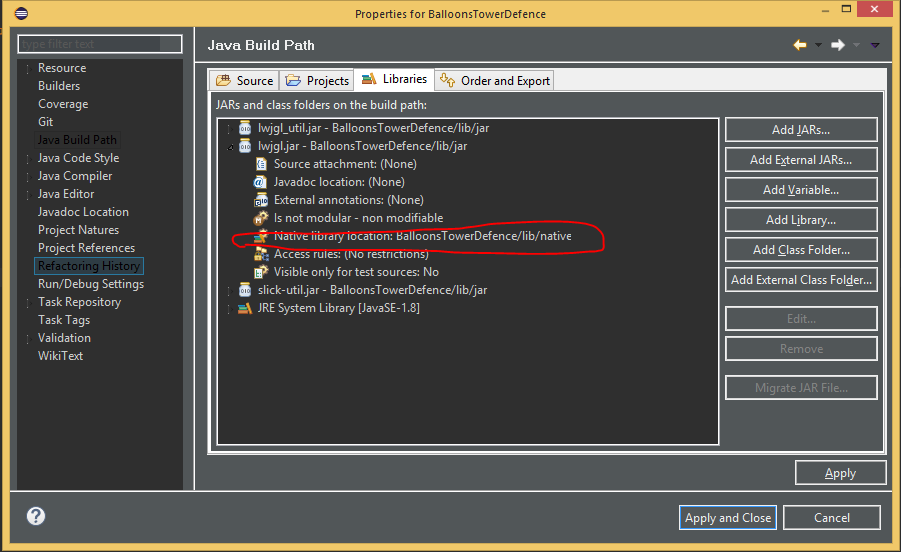
## Files included

All these files are included in the lib folder in the project source

### natives

* All natives are included for the project in the lib/natives folder

### all natives will need to be asinged to lwjgl.jar file



If you import this project or clone it from the Github repository the natives and all ref libs will be in place for you, or if you clone it, and then import the project from the .zip file.

## Core gameply

The game is mostly based on the ever popular game Bloons Tower Defense created by Ninja Kiwi

The game reflects some of the concepts presented by its predecessor in which it has:

* Place able towers which have different strengths and abilities.
* A system in which you may create your own map and paths for the balloons to travel on.
  + There are some know issues that may affect the gameplay experience (listed later in this document).
* A loading screen (Its just there for help with future development, more on this later)
* Ability to create custom towers (for now it is only possible through editing the source).
* Different balloon types with different heath and speeds.

### Potential exceptions and problems

* Though my expectations were merely reached through some of the implementation of concepts that produced high end game design (i.e. multi-threading though I could not find a place to use this at, more on this later).
* Some computers may not be able to handle the calculations required in this game if there is a medium amount of load on the system, caused by background tasks like Chrome and Photoshop.
* Sometimes the towers will glitch and not shoot the balloons

### Changes made from initial project outline

In the following table, I have listed all the changes I have made to the project since the initial outline

## Classes

### Package: balloonsTowerdefence

1. [Balloon.java](#balloonjava)
2. BalloonBlue.java
3. BalloonRed.java
4. Dart.java
5. DartLazer.java
6. DartNinjaStar.java
7. DartNormal.java
8. DartType.java
9. Entity.java
10. Explosion.java
11. Floor.java
12. FloorGrid.java
13. FloorType.java
14. Game.java
15. HighScore.java
16. Instructions.java
17. Leaderboard.java
18. LevelEditor.java
19. MainMenu.java
20. MakeFrame.java
21. MonkeyTower.java
22. MonkeyTowerDartMonkey.java
23. MonkeyTowerIceMonkey.java
24. MonkeyTowerNinjaMonkey.java
25. MonkeyTowerSuperMonkey.java
26. MonkeyTowerType.java
27. Player.java
28. Round.java
29. RoundManager.java
30. SplashScreenLoading.java
31. StartGame.java
32. TestScreen.java
33. TurningPoint.java

### Package: handlers

1. Informal.java
2. Error.java

### Package: other

1. DrawInFrame.java
2. ImageTools.java
3. LevelToolKit.java
4. Sprite.java
5. StateManager.java
6. Timer.java

### Package: userInterface

1. Button.java
2. Image.java
3. Label.java
4. ProgressBar.java
5. Slider.java
6. UserInterFace.java

## Goals

### completed

1. Make place able towers
2. Make a level editor
3. An algorithm to traverse a path
4. Basic framework for user interface and game mechanics

### Unfinished

1. Leader boards and user saving
2. Select map screen
3. Select difficulty/game mode screen

CLASS METHOD DOCUMENTATION

BALLOON.java

Description:

This class is the super object of all other balloon classes (BalloonRed etc).

It controls most of the tasks needed to perform for the balloon to render on to the screen and is the heart of the balloon framework. Like movement and health, hits and drawing and most importantly rewarding the player money if the balloon is in fact popped.

# Imports:

### STATIC:

* other.DrawInFrame.DrawQuadWithTexture
* other.DrawInFrame.GRID\_SQUARE\_SIZE
* other.DrawInFrame.HEIGHT
* other.DrawInFrame.LoadTexture
* other.DrawInFrame.WIDTH
* other.Timer.Delta

## NOn-static

* java.util.ArrayList
* org.newdawn.slick.opengl.Texture

### IMPLEMENTS:

* Entity (Interface)

Variables:

* private static final int MONEY\_FOR\_KILL = 10;
* private int width, height, moneyReward, currentTurnPoint;
* private float speed, x, y, health, startingHealth, hiddenHealth;
* private Texture instanceTexture;
* public BalloonType type;
* private Floor startingFloorObject;
* private boolean first, alive;
* private FloorGrid grid;
* private boolean calledChangeMoney = false;
* private ArrayList<TurningPoint> turnigPoints;
* private int[] directions; // this contains the x and y momentum

# MEthods:

Balloon(int, int, FloorGrid, BalloonType)

Description:

constructor a balloon object is created with the following as parameters int

floorX, int floorY, FloorGrid grid, BalloonType type

the balloon is spawned on the floorX and floorY coordinates and the default

values provided by the type specified

Balloon(Texture, Floor, FloorGrid, int, int, float, float)

Description:

constructor a balloon object is created with the following as parameters

Texture instanceTexture, Floor startingFloorObject, FloorGrid grid, int

width, int height, float speed, float health

the balloon is created with a texture and spawned on the startingFloorObject

with a width and height and speed and health

tick()

Description:

tick is called every loop of the main game loop to update the current factors

of the balloons position and checks if the balloon is alive and gets the next

turning point and moves the balloon depending on the speed and the directions

array

isTurningPointReached()

Description:

returns true if the turning point has been reached

endReached()

Description:

This pops the balloon if it makes it through the entire way without beeing

poped and subtracts a life from the player

damage(int)

Description:

damages the balloon object and requires a damage parameter to subtract from

the health if the health is 0 or less then added the specified amount of

money to the player object

findDirection(Floor)

Description:

findDirection returns a int array which contains two values represent if the

balloon should move in either axis

post: returns a direction array that contains values either 1, 2, 0, -1

representing movement if the value is 1 then the balloon moves by

mulltiplieing the speed and delta by 1 otherwise it returns 2 if the end is

reached and 0 if the movement is not required in either the x or y axis also

it is -1 one if it is going backwards or against the axis in this case if it

is -1 on the 1st index then it is going up on the y axis

fillTurningPointArrayList()

Description:

Fills the array list of turning point objects containing all directional

object for maneuvering the balloon object it loops through util it finds the

end of the path using a basic guess and check algorithm path finding

algorithm,

post: the turning point list is filled with all the turning points discovered

in the map

findTurningPoint(Floor, int[])

Description:

Returns an object representing the current turning point by looping through

all possibilites of the checkpoint's location up, down, left, right from the

current location starting floor is the object currently on.

In summary this algorithm is more complicated to explain then to make it

Basically checks if there is a turning point in the direction the balloon is

Going by looking at the dir array passed as a parameter and then sets found

To true if the turning point is found. For example if the x axis movement is

0 and the y axis is 1 then this algorithm checks the floor down 1 times the

Counter so the first one is just 1 down and the next loop the counter is 2

And so it checks the 2nd floor down and if it is a turning point then found

Is set to true and the loop is broken I found some tutorial talking about

This and provided some pseudo code but I couldn't find it again so I am sorry

I can't reference it here

draw()

Draws the object on the screen represented by the texture on the screen using

a static method from DrawInFrame

kill()

sets alive to false so that the balloon is considered dead and is removed

from the round

reduceHiddenHealth(float)

reduces the health of the balloon and requires a amount to do by so.

isAlive()

returns the isAlive variable

# 