TouhouBomberGirls

1. Overview

TouhouBomberGirls is a game that has 2 players. Each player has only objective that is to kill other player. When the game start there will be a screen(shown in figure 1)with background music. After that, players can select to play or to read how to play first(Shown in figure 2). In How to play there are 2 pages if you want to read another page you can click Next button(in Figure 2). Then, the screen will show How to play page 2(Shown in figure 3). After that, you can click Let's go to play game or Good Bye to exit from the game. When the game end, there will be dialog show up(Shown in figure 6). After players click OK, the game will exit automatically.

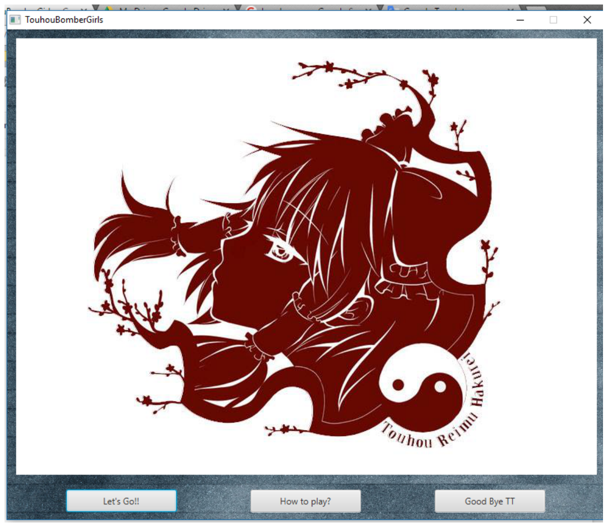
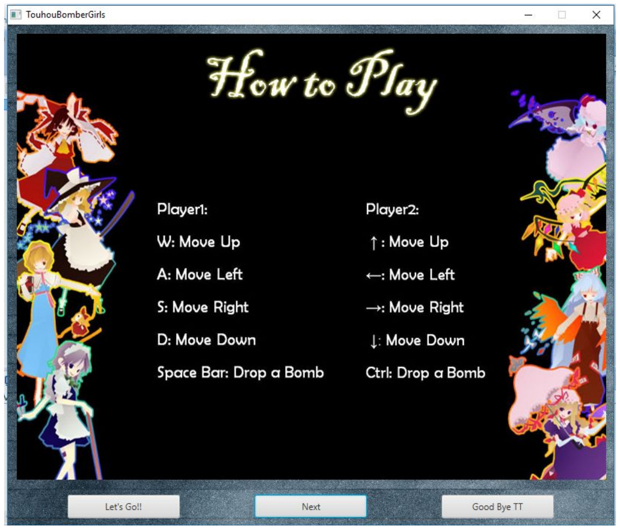


Figure 1(Initial Screen)



Player1:	Player2:
W: Move Up	↑: Move Up
A: Move Left	←: Move Left
S: Move Right	→: Move Right
D: Move Down	↓: Move Down
Space Bar: Drop a Bomb	Ctrl: Drop a Bomb

Figure 2(How to Play 1)

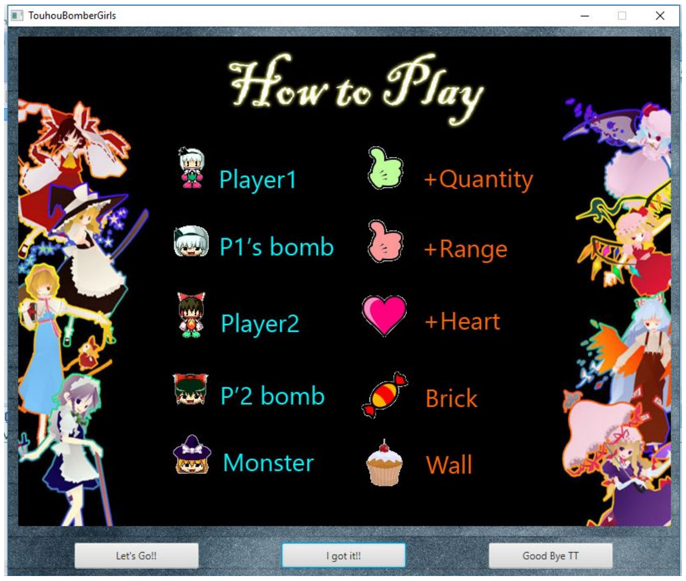


Figure 3(How to Play 2)

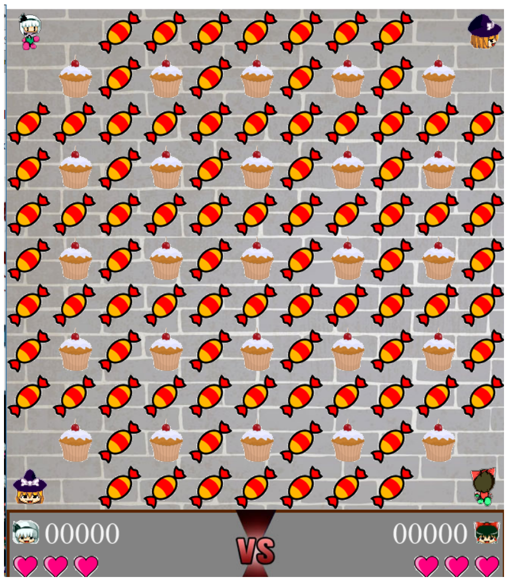


Figure 4(Start Game Screen)

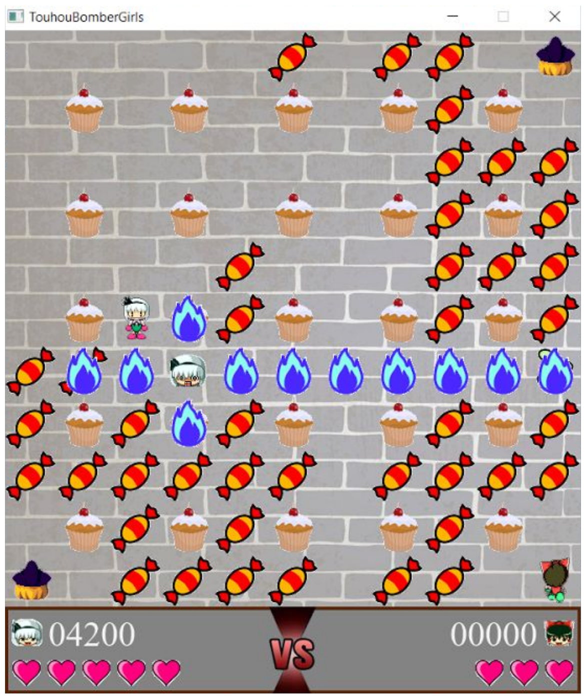
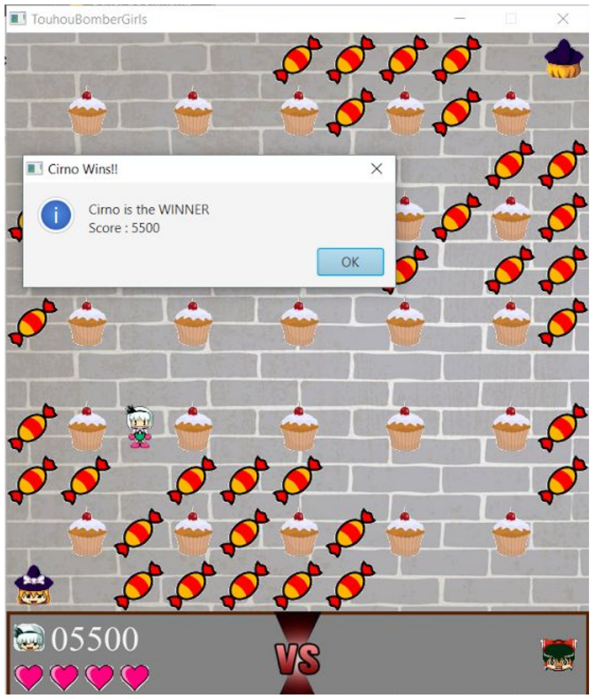


Figure 5(Bomb explode)



Cirno Wins!!

Cirno is the WINNER
Score : 5500

Figure 6(Dialog end game)

The following are the game components and their conditions.

1) Player

- There are 2 players that are Player1 ( ) and Player 2 ( ).
- Player 1 name Cirno and Player 2 name Aya.
- At the beginning, each player will have 3 heart, 1 bomb and 1 bomb range.
- Each player can gain score from destroy candy,enemy and another player.
- if there is a player who die first, another player will win automatically.
- if 2 players die at the same time, it will be a draw.

2) Enemy ( )

- There are 2 enemy at the right top and left bottom.
- Each enemy will move randomly.
- Each enemy can kill player in 1 hit.
- Give 500 score for player who kill.

3) Candy ( )

- Can be destroyed
-When candy was destroyed, it will randomly generate Item.
- Give 100 score for player who destroy.

4) Cake ( )

- Can't be destroyed.

5) Bomb

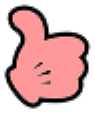
- Each player has his own bomb style. (shown in figure 3)

Player 1 : Bomb( ) Flame( )

Player 2 : Bomb( ) Flame( )

- Bomb will explode range depend on player's bomb range.
- Bomb can be placed depend on player's bomb.
- Bomb can damage player for 1 heart and can kill enemy or candy in 1 hit.
- Bomb can destroy Item(Carefully when place bomb).

6) Item

- There are 3 Item in the game which are Quantity,Range and Heart. (shown in figure 3)
- Quantity Item ( ) will give another bomb to player who collect.
- Range Item () will give +1 to bomb range of player who collect.
- Heart Item ( ) will give +1 Heart to player who collect.(Max is 5)
- Spawn randomly when candy is destroyed(Spawn at the place where candy is destroyed).
- Item can be ruin by bomb.

For the size of the screen will be following the figure below

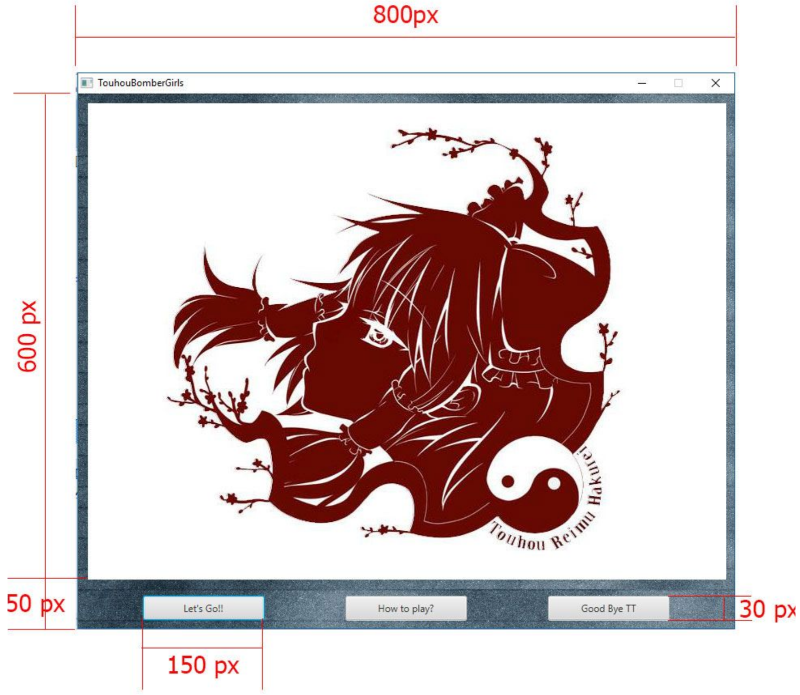


Figure 7(Measurement of initial Screen)

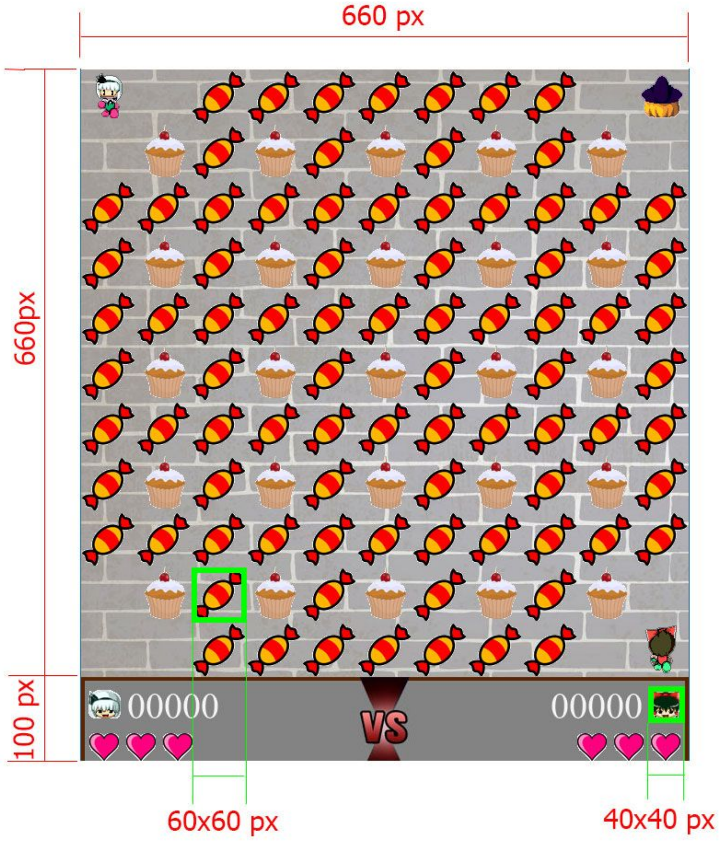
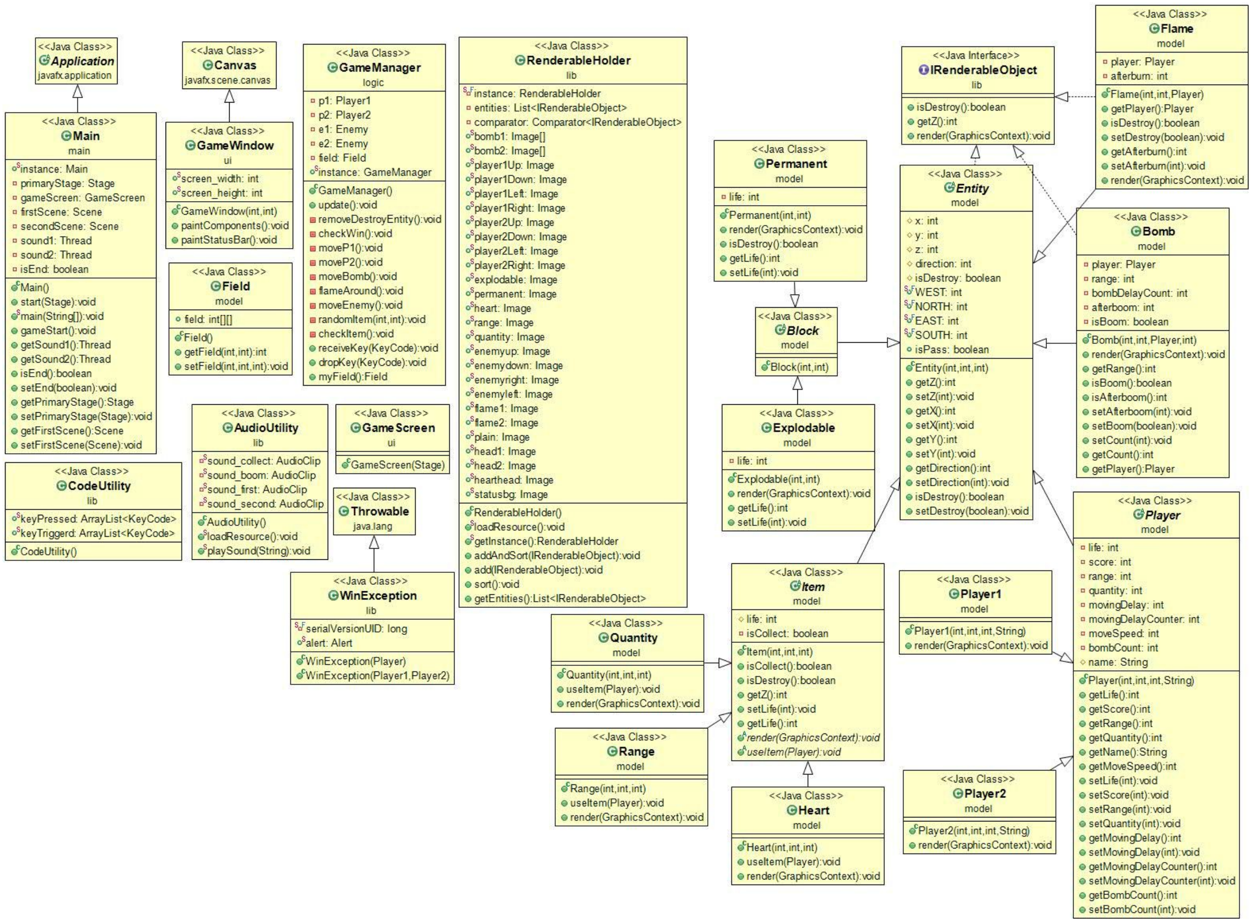


Figure 8(Measurement of Game Screen)

2. UML



<<Java Class>> Application javafx.application
<<Java Class>> Main main
instance: Main
primaryStage: Stage
gameScreen: GameScreen
firstScene: Scene
secondScene: Scene
sound1: Thread
sound2: Thread
isEnd: boolean
Main()
start(Stage):void
main(String[]):void
gameStart():void
getSound1():Thread
getSound2():Thread
isEnd():boolean
setEnd(boolean):void
getPrimaryStage():Stage
setPrimaryStage(Stage):void
getFirstScene():Scene
setFirstScene(Scene):void
<<Java Class>> CodeUtility lib
keyPressed: ArrayList<KeyCode>
keyTriggerd: ArrayList<KeyCode>
CodeUtility()
<<Java Class>> Canvas javafx.scene.canvas
<<Java Class>> GameWindow ui
screen_width: int
screen_height: int
GameWindow(int,int)
paintComponents():void
paintStatusBar():void
<<Java Class>> Field model
field: int[][]
Field()
getField(int,int):int
setField(int,int,int):void
<<Java Class>> AudioUtility lib
sound_collect: AudioClip
sound_boom: AudioClip
sound_first: AudioClip
sound_second: AudioClip
AudioUtility()
loadResource():void
playSound(String):void
<<Java Class>> GameManager logic
p1: Player1
p2: Player2
e1: Enemy
e2: Enemy
field: Field
instance: GameManager
GameManager()
update():void
removeDestroyEntity():void
checkWin():void
moveP1():void
moveP2():void
moveBomb():void
flameAround():void
moveEnemy():void
randomItem(int,int):void
checkItem():void
receiveKey(KeyCode):void
dropKey(KeyCode):void
myField():Field
<<Java Class>> GameScreen ui
GameScreen(Stage)
<<Java Class>> Throwable java.lang
<<Java Class>> WinException lib
serialVersionUID: long
alert: Alert
WinException(Player)
WinException(Player1,Player2)
<<Java Class>> RenderableHolder lib
instance: RenderableHolder
entities: List<IRenderableObject>
comparator: Comparator<IRenderableObject>
bomb1: Image[]
bomb2: Image[]
player1Up: Image
player1Down: Image
player1Left: Image
player1Right: Image
player2Up: Image
player2Down: Image
player2Left: Image
player2Right: Image
explodable: Image
permanent: Image
heart: Image
range: Image
quantity: Image
enemyup: Image
enemydown: Image
enemyright: Image
enemyleft: Image
flame1: Image
flame2: Image
plain: Image
head1: Image
head2: Image
hearthead: Image
statusbg: Image
RenderableHolder()
loadResource():void
getInstance():RenderableHolder
addAndSort(IRenderableObject):void
add(IRenderableObject):void
sort():void
getEntities():List<IRenderableObject>
<<Java Class>> Permanent model
life: int
Permanent(int,int)
render(GraphicsContext):void
isDestroy():boolean
getLife():int
setLife(int):void
<<Java Class>> Block model
Block(int,int)
<<Java Class>> Explodable model
life: int
Explodable(int,int)
render(GraphicsContext):void
getLife():int
setLife(int):void
<<Java Class>> Quantity model
Quantity(int,int,int)
useItem(Player):void
render(GraphicsContext):void
<<Java Class>> Range model
Range(int,int,int)
useItem(Player):void
render(GraphicsContext):void
<<Java Class>> Item model
life: int
isCollect: boolean
Item(int,int,int)
isCollect():boolean
isDestroy():boolean
getZ():int
setLife(int):void
getLife():int
render(GraphicsContext):void
useItem(Player):void
<<Java Class>> Heart model
Heart(int,int,int)
useItem(Player):void
render(GraphicsContext):void
<<Java Interface>> IRenderableObject lib
isDestroy():boolean
getZ():int
render(GraphicsContext):void
<<Java Class>> Entity model
x: int
y: int
z: int
direction: int
isDestroy: boolean
WEST: int
NORTH: int
EAST: int
SOUTH: int
isPass: boolean
Entity(int,int,int)
getZ():int
setZ(int):void
getX():int
setX(int):void
getY():int
setY(int):void
getDirection():int
setDirection(int):void
isDestroy():boolean
setDestroy(boolean):void
<<Java Class>> Player1 model
Player1(int,int,int,String)
render(GraphicsContext):void
<<Java Class>> Player2 model
Player2(int,int,int,String)
render(GraphicsContext):void
<<Java Class>> Flame model
player: Player
afterburn: int
Flame(int,int,Player)
getPlayer():Player
isDestroy():boolean
setDestroy(boolean):void
getAfterburn():int
setAfterburn(int):void
render(GraphicsContext):void
<<Java Class>> Bomb model
player: Player
range: int
bombDelayCount: int
afterboom: int
isBoom: boolean
Bomb(int,int,Player,int)
render(GraphicsContext):void
getRange():int
isBoom():boolean
isAfterboom():int
setAfterboom(int):void
setBoom(boolean):void
setCount(int):void
getCount():int
getPlayer():Player
<<Java Class>> Player model
life: int
score: int
range: int
quantity: int
movingDelay: int
movingDelayCounter: int
moveSpeed: int
bombCount: int
name: String
Player(int,int,int,String)
getLife():int
getScore():int
getRange():int
getQuantity():int
getName():String
getMoveSpeed():int
setLife(int):void
setScore(int):void
setRange(int):void
setQuantity(int):void
getMovingDelay():int
setMovingDelay(int):void
getMovingDelayCounter():int
setMovingDelayCounter(int):void
getBombCount():int
setBombCount(int):void

3. Class Details

3.1 Class lib.AudioUtility

This class is used for collect all game's sound.

3.1.1 Field

-static Audioclip sound_collect	Sound when player collect item.
-static Audioclip sound_boom	Sound when bomb is exploded.
-static Audioclip sound_first	Sound when run application.
-static Audioclip sound_second	Sound when start a game.

3.1.2 Method

+void loadResource()	A static way to load resource only one time. -load collect.wav to sound_collect, boom.wav to sound_boom, op.wav to sound_first, naturesounds.wav to sound_second
+void playSound(String identifier)	A static method that identify string and let the right sound plays.

3.2 Class lib.CodeUtility

This class is used for collect input key from keyboard.

3.2.1 Field

+ArrayList<KeyCode> keyPressed	List of a key which is still pressed
+ArrayList<KeyCode> keyTriggered	List of a key which just triggered

3.3 Class lib.IRenderableObject

Any class that need to draw image must implement this interface

3.3.1 Method

+int getZ()	Return the priority for drawing.
+void render(GraphicContext gc)	Implement drawing.
+boolean isDestroy()	Return whether to draw this object or not.

3.4 Class lib.RenderableHolder

A share collection of renderable objects

3.4.1 Field

-List<IRenderable> entities	List of IRenderable objects, this list mustalways be sorted
-Comparator<IRenderable> comparator	A Comparator to sort entities list by z value
+Image[] bomb1, bomb2	Static image of bomb's effect.
+Image player1Up, player1Down, player1Left, player1Right, player2Up, player2Down, player2Left,player2Right, explodable, permanent, heart, range, quantity, enemyup, enemydown, enemyright, enemyleft,flame1, flame2, plain, head1, head2, hearthead, statusbg;	Static image of all picture except bomb.
+final RenderableHolder instance	A Singleton object of RenderableHolder

3.4.2 Constructor

+RenderableHolder()	Initialize -entities -bomb1 , bomb2 with size 2 -compatator should be compare base on z value of the IRenderable object

3.4.3 Method

+void loadRasourec()	A static way to load resource only one time. Load all image to Variable that have same name(let p mean player). For bomb -bomb1c1 to bomb1[0] -bomb1c2 to bomb1[1] -bomb2c1 to bomb2[0] -bomb2c2 to bomb2[1]
+RenderableHolder getInstance()	Static way to get instance
+voidaddAndSort (IRenderableObject entity)	Call add and sort method
+void add(IRenderableObject entity)	Add entity into entities
+void sort()	Use for sort item in entities
+synchronized List <IRenderableObject> getEntities()	Return entities

3.5 Class lib.WinException

This class use for throw exception when game end.

3.5.1 Field

+final long serialVersionUID	A static variable equal to 1L
+Alert alert	A static alert box

3.5.2 Constructor

+WinException(Player player)	Let alert be an information alert that if player1 win, show the message that player1 win and score.Also as a player2.Then exit game when click OK button.
+WinException(Player1 p1, Player2 p2)	Let alert be an information alert that show that game is draw and show score of both of players. Then exit game when click OK button.

3.6 Class main.Main

This class is a JavaFX Application responsible for showing user interface

3.6.1 Field

+Main instance	A singleton instance of this class
-Stage primarystage	For set primarystage
-GameScreen gameScreen	For set gamescreen
-Scene firstScene, secondScene	For set scene
-Thread sound1, sound2	For let sound play
-boolean isEnd	For check that is game end? Set to false.

3.6.2 Method

+void start(Stage primaryStage) throws Exception	Initialize the start interface -use gamescreen as scene Let thread sound1 play op sound.
+void gameStart()	Set new scene with width=660 and height=760 and set gamewindow be a scene. Let thread sound1 play nature sound. Set event to receiveKey And dropKey game's control key.(See at Figure 4.1) Call update ,paintComponents and paint StatusBar.

+Getter for sound1,sound2	
+Getter&Setter for isEnd, primarystate and firstScene	

3.7 Class model.Entity

This class represent base class for all renderable objects.

3.7.1 Field

#int x,y	Position of object on screen
#int z	Drawing priority for this objective.
#int direction	Show direction of this object
#isDestroy	Check that this object is destroyed or not.
+boolean isPass	Check that is it can pass Set it to true
+final int WEST = 0, NORTH = 1, EAST = 2, SOUTH = 3	Static direction

3.7.2 Constructor

+Entity(int x, int y, int direction)	Initialize its position, direction and set isDestroy to false.

3.7.3 Method

+Getter&Setter for x, y	For setter if value is less than 0 set it to 0 and if it more than 600 set it to 600
+Getter&Setter for direction, isDestroy, z	

3.8 Class model.Block

This class represent base class for block.

3.8.1 Constructor

+Block(int x, int y)	Initialize position ,direction = south and isPass = false

3.9 Class model.Explodable

This class represent block that can destroy.

3.9.1 Field

-int life	This object’s life

3.9.2 Constructor

Explodable(int x, int y)	Initialize position ,z=1 and setlife to 1

3.9.3 Method

+Getter&Setter for life	When life is 0, block will be destroyed
+void render(GraphicsContext gc)	Draw explodable at its position

3.10 Class model.Permanent

This class represent block that cannot destroy.

3.10.1 Field

-int life	This object's life

3.10.2 Constructor

Permanent(int x, int y)	Initialize position ,z=1 and setlife to max value

3.10.3 Method

+Getter&Setter for life	
+void render(GraphicsContext gc)	Draw permanent at its position

3.11 Class model.Bomb

This class represent bomb

3.11.1 Field

-Player player	Bomb's owner
-int range	Bomb's range
-int bombDelayCount	Bomb's tick
-boolean afterboom	Check bomb after boom's status
-boolean isBoom	Check that bomb is waiting or bomb. Set it to false.

3.11.2 Constructor

Bomb(int x, int y, Player player, int range)	Initialize all variable and direction is south

3.11.3 Method

+void render(GraphicsContext gc)	Check bomb's owner and check isBoom Bomb1[] for player1 Bomb2[] for player2 Index 0 is didn't boom Index 1 is boom
+Getter for player	
+Getter&Setter for range, afterboom, isBoom, bombDelayCount	

3.12 Class model.Item

This class represent base class of item object

3.12.1 Field

#int life	Item's life
-boolean isCollect	Check item is collected or not

3.12.2 Constructor

+Item(int x, int y, int direction)	Initialize position, direction ,z=1 ,life = 3 and isCollect = false

3.12.3 Method

+Getter for isCollect, isDestroy,z	
+Getter&Setter for life	When life less than 1, it will destroyed
+abstract void render(GraphicsContext gc)	
+abstract void useItem(Player p)	

3.13 Class model.Heart

This class represent Heart item for heal

3.13.1 Constructor

+Heart(int x, int y, int direction)	Initialize position, direction

3.13.2 Method

+void useItem(Player p)	When this didn't destroyed ,increase life by 1, destroy this and play collect sound
+void render(GraphicsContext gc)	Draw heart at its position

3.14 Class model.Quantity

This class represent Quantity item so player can place more bomb.

3.15.1 Constructor

+Quantity(int x, int y, int direction)	Initialize position, direction

3.15.2 Method

+void useItem(Player p)	When this didn't destroyed ,increase 1 quantity, destroy this and play collect sound
+void render(GraphicsContext gc)	Draw quantity at its position

3.15 Class model.Range

This class represent Range item so bomb can bomb further.

3.15.1 Constructor

+Range(int x, int y, int direction)	Initialize position, direction

3.15.2 Method

+void useItem(Player p)	When this didn't destroyed ,increase 1 range, destroy this and play collect sound
+void render(GraphicsContext gc)	Draw range at its position

3.16 Class model.Field

Since we have game canvas size 660*660 so we split it into 11*11 block and this class show that each block can pass(0) or not(1).

3.16.1 Field

+int[][] field	Array that contain type of block

3.16.2 Constructor

+Field()	Set all value in array to 0

3.16.3 Method

+Getter&Setter for field	

3.17 Class model.Player

This class represent base class of player

3.17.1 Field

-int life, score, range, quantity, movingDelay	Parameter for player
-int movingDelayCounter	Set it to 1
-int moveSpeed	Set it to 60
-int bombCount	Set it to 0
#String name	player's name

3.17.2 Constructor

+Player(int x, int y, int direction, String name)	Initialize position, direction and name of player and set life to 3, score to 0 , range and quantity to 1 and isDestroy to flase

3.17.2 Method

+Getter&Setter for life, score, range, quantity, movingDelay, movingDelayCounter, bombCount	
+Getter for name and moveSpeed	

3.18 Class model.Player1

This class represent game's player1

3.18.1 Constructor

+Player1(int x, int y, int direction, String name)	Initialize position, direction and name

3.18.2 Method

+void render(GraphicsContext gc)	Draw image up to its direction -north :player1Up -east :player1Right -west :player1Left -south :player1Down

3.19 Class model.Player2

This class represent game's player2

3.19.1 Constructor

+Player2(int x, int y, int direction, String name)	Initialize position, direction and name

3.18.2 Method

+void render(GraphicsContext gc)	Draw image up to its direction -north :player2Up -east :player2Right -west :player2Left -south :player2Down

3.19 Class model.Enemy

This class represent enemy of player in game

3.19.1 Field

-int moveDelayCounter	
-final int moveDelay	Enemy move delay and set it to 5

3.19.2 Constructor

+Enemy(int x, int y, int direction)	Initialize position and direction

3.19.3 Method

+render(GraphicsContext gc)	Draw image up to its direction -north :enemyUp -east :enemyRight -west :enemyLeft -south :enemyDown
+void move()	Enemy has 80% chance to move upon its direction and 20% chance to random its direction. If it can't move then random its direction
+void attackPlayer(Player player)	Let's player die instantly

3.20 Class ui.GameScreen

This class represent first screen of this game

3.20.1 Constructor

+GameScreen(Stage stage)	Initialize GUI -set size to 800*600 -draw bg2 -create gridpane size 800*50 to store 3 buttons size 150*30 each For action on button 1.click to start game 2.each click switch screen to instruction > instruction2 > bg 3.click to exit game

3.21 Class ui.GameWindow

This class use for draw everything when game plays

3.21.1 Field

+static int screen_width, screen_height	Size of game screen

3.21.2 Constructor

+GameWindow(int width,int height)	Initialize size of screen

3.21.3 Method

+void paintComponents()	Set size of screen Draw plain image & everything in RenderableHolder
+void paintStatusbar()	This start at position 0,660 Draw head1,head2 image and score at top and Draw life of each player like Figure4. If someone die, draw only bomb1[1] or bomb2[1] like Figure 6.

3.22 Class logic.GameManager

This class contain all logic that use in this game.

3.22.1 Field

+static GameManeger instance	A singleton instance of this class
-Player1 p1	Game's player1
-Player2 p2	Game's player2
-Enemy e1,e2	Game's enemy
-Field field	Represent status game's field.

3.22.2 Constructor

+GameManager()	Initialize -field, instance. -e1 at "600,0,south", e2 at "0,600,north" -p1 at "0,0,south" name Cirno and p2 at "600,600,north" name Aya. -wall and block like Figure 4 and set Field to 1 -add all player, enemy, block and wall to RenderableHolder

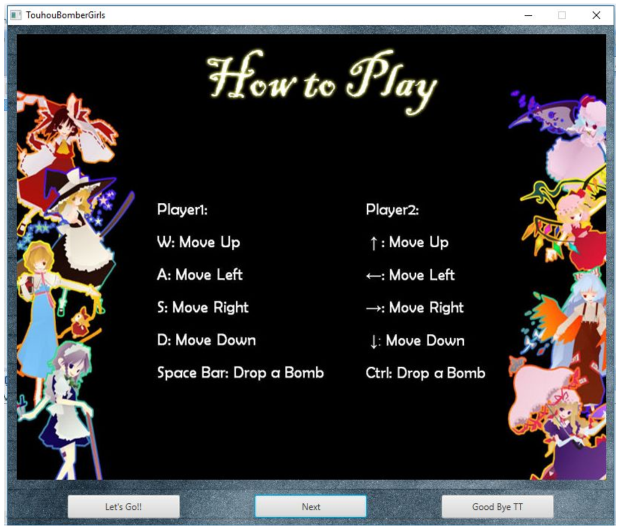
3.22.3 Method

+void update()	Call -moveP1() -moveP2() -moveBomb() -flameAround() -moveEnemy() -checkItem() -checkWin() -removeDestroyEntity()
-void removeDestroyEntity()	Remove all destroyed object and set field of that object to 0 if that object is Explodable
-void checkWin()	If someone is destroyed set isEnd to true and throw WinException with the winner. If both of them die throw with both of them
-moveP1(),moveP2()	Check input in keyPress(figure2) then set direction. Check that it can move to next step if it can then move it upon its moveSpeed. For bomb key it can lay only 1 bomb at one click(use keyTrigger) and can't lay bomb more than its quantity.
-moveBomb()	For all bomb in renderableHolder check if afterboom =1, then set field to 0 and decrease bombCount by 1 and play boom sound But afterboom =0 check that bomb count equal to 12 or not.If it's equal increase afterboom by 1 and create flame in 4 direction around upon bomb's owner range.but if flame struct with wall then it's stop.(Figure 5)
-move flameAround()	For all flame in RenderableHolder if afterburn=1 destroy it and check that is it on same position with player, explodable, enemy, item or not. If with player, decrease player's life by 1 and if life = 0 other player get 2000 scores. If with explodable, decrease explodable's life by 1 then flame's owner get 100 scores and random item there. If with enemy, enemy die and flame's owner get 500 scores. If with item, decrease item's life by 1. But if afterburn=0, increase it by 1.
-void moveEnemy()	If e1 is alive call move and check that is it stay on same position with player if yes, attack player. e2 too.

-void randomItem(int x, int y)	10% chance to create item Range 10% chance to create item Quantity 10% chance to create item Heart
-void checkItem()	Check all item in RenderableHolder that is it on same position with player.If yes, call useitem
+void receiveKey(KeyCode new_code)	If keyPressed didn't contain received key, add it to keyPressed
+void dropKey(KeyCode new_code)	If keyPressed contain received key, remove it from keyPressed and keyTriggered
+Field myField()	Return field

4. How to Play

When the game is started. Each player has to destroy candy to gather score and item. If player can kill enemy then the player will be given 600 score. To win is to kill another player.



Player1:	Player2:
W: Move Up	↑: Move Up
A: Move Left	←: Move Left
S: Move Right	→: Move Right
D: Move Down	↓: Move Down
Space Bar: Drop a Bomb	Ctrl: Drop a Bomb

Figure 4.1

Each player can control the character by pressing the key above(Player 1 is at the left top and Player 2 is at the right bottom of the game screen). When candy is destroyed, there will be a chance for item to be spawned(shown in figure 4.2)

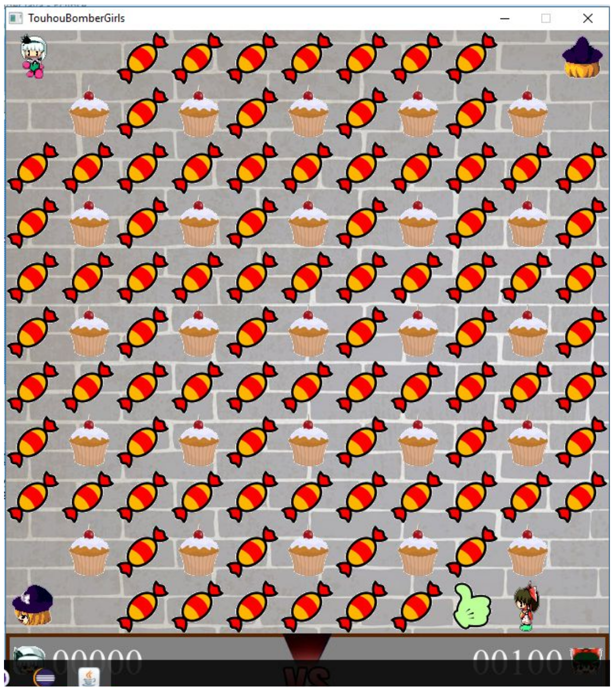
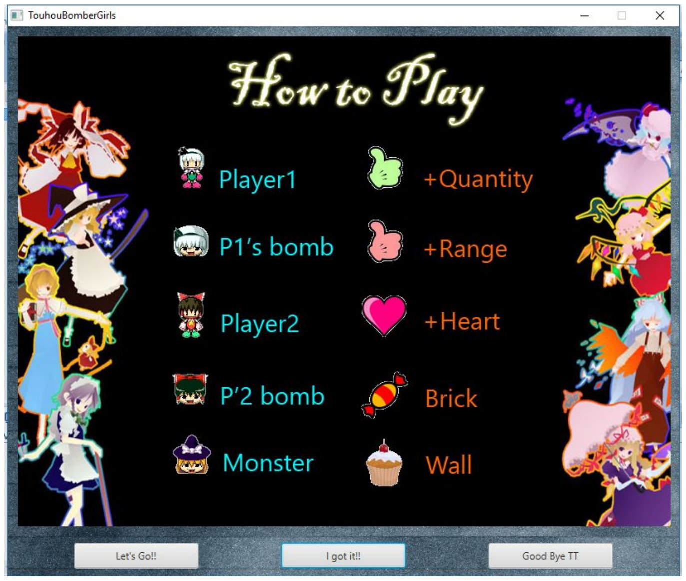


Figure 4.2

Each item has its own symbol(shown in figure 4.3)



This is all about how to play Touhou bombergirls. Wish you have a lot of fun.