Bomberman

Created by

Nonya Kangwanteerawat

2110215 Programming Methodology
Semester 2 Year 2019 Chulalongkorn
University

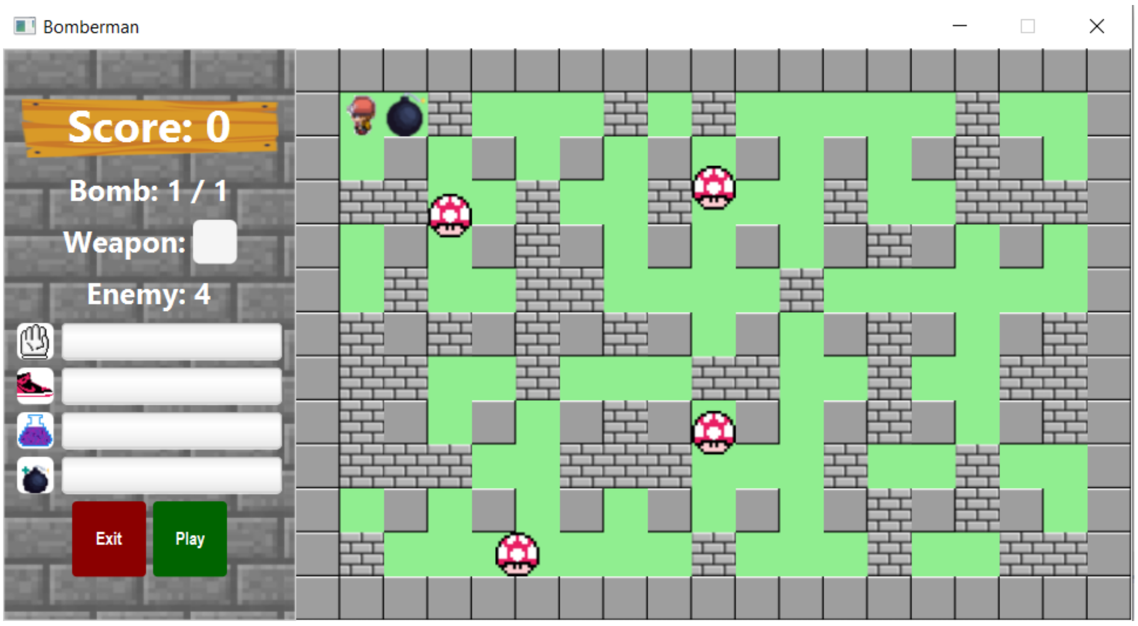
Bomberman

Introduction

Bomberman is a strategic maze-based game. The player must defeat enemies by bombs and reach an exit to progress through levels.

Gameplay

The player needs to strategically place down the bombs, which will explode in four directions after certain amount of time, in order to destroy obstacles and enemies. There are power-ups and weapons which will give player benefits. Game Objective is to kill all enemies and get through the door hidden behind the wall. If you beat the game, the score you get will be the number of coins you get. You can use the coins to upgrade your base stats and buy a weapon.

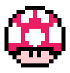


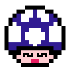
Controlling

Use **SPACE** to place the bombs or to use the weapon and use **W**, **A**, **S**, **D** to move UP, LEFT, DOWN, RIGHT, respectively.

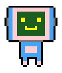
Enemies

There are 5 types of enemies which are Mushroom, Blind Mushroom, Mechabomb, Bomberbot and Bomb Eater.


Mushroom is slow and able to follow the player.


Blind Mushroom is fast, but he can't follow you.

Mechabomb can transform to a bomb.


Bomberbot can place a bomb.

Bomb Eater can eat the bomb. He will follow the bombs.

Power-ups

There are 4 types of power-ups in this game. They will improve player's stats which are speed, maximum bombs, bomb power and gloves.

The player can use gloves to push a bomb.

This will increase speed.

This will increase maximum bombs.

This will increase bomb power.

Weapons

There are 3 types of weapons in this game, which are Land Mine, Rocket Launcher and Flamethrower.

This is **Land Mine**. Player can use this weapon to plant land mine, and it will explode if enemy pass through.

This is **Rocket Launcher**. Player can use this to fire the rocket into enemies or obstacles.

This is **Flamethrower**. Player can use this to destroy enemies or obstacles.

Objects

This wall can't be destroyed.

This wall can't be destroyed. It will appear only if you run into the wall.

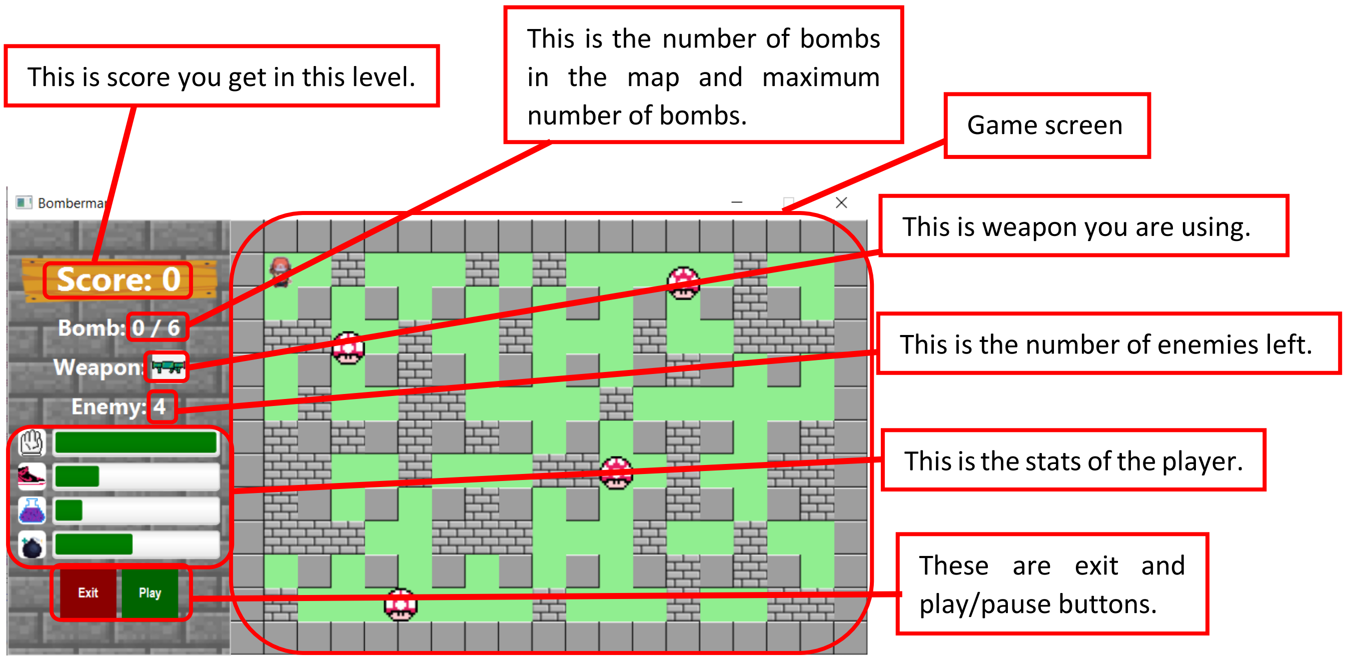
This wall can be destroyed by bombs, flamethrower and rocket.

This Bomb will be exploded if interact with blast.

This Bomb is from an enemy. It won't be exploded if interact with blast.

This is a door. You must get through this door in order to pass the level.

GameScene



This is score you get in this level.

This is the number of bombs in the map and maximum number of bombs.

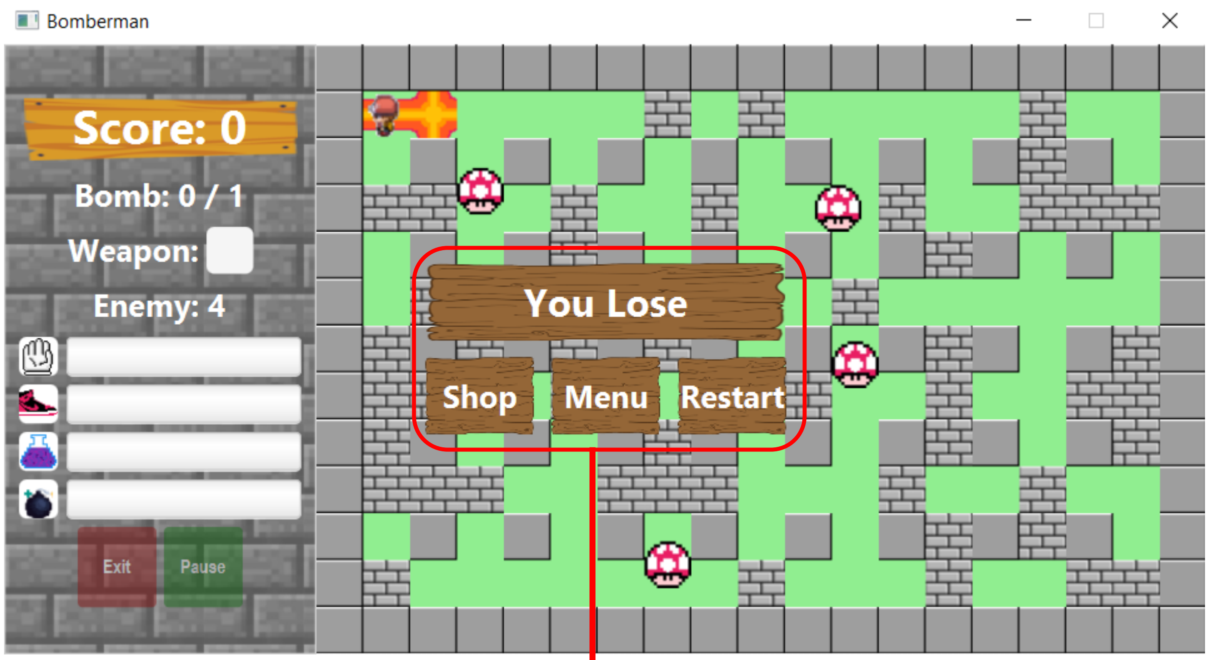
Game screen

This is weapon you are using.

This is the number of enemies left.

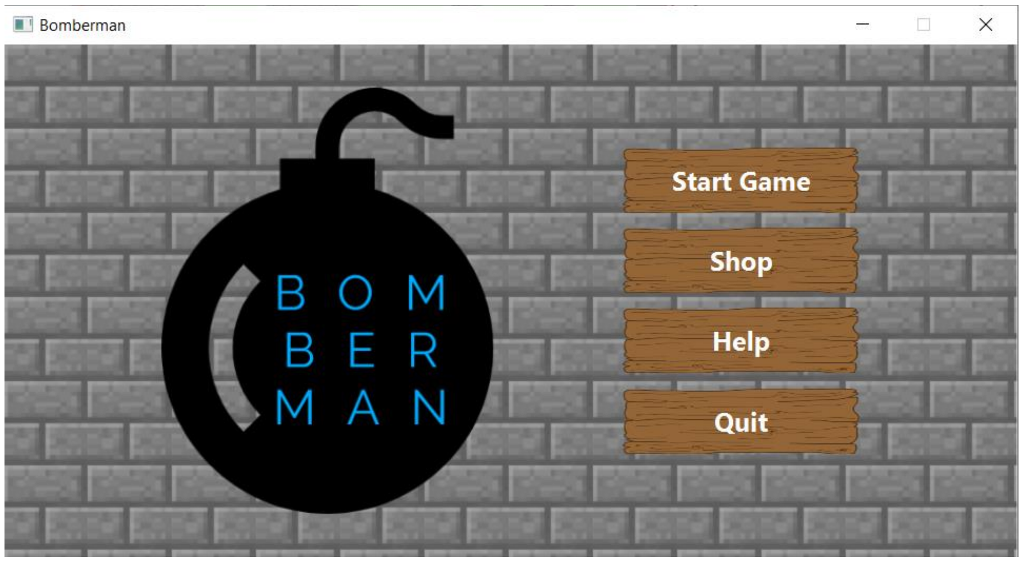
This is the stats of the player.

These are exit and play/pause buttons.



These buttons will be shown when the game is over.

MainMenuScene



There are 4 buttons, which are Start Game, Shop, Help and Quit, in this scene.

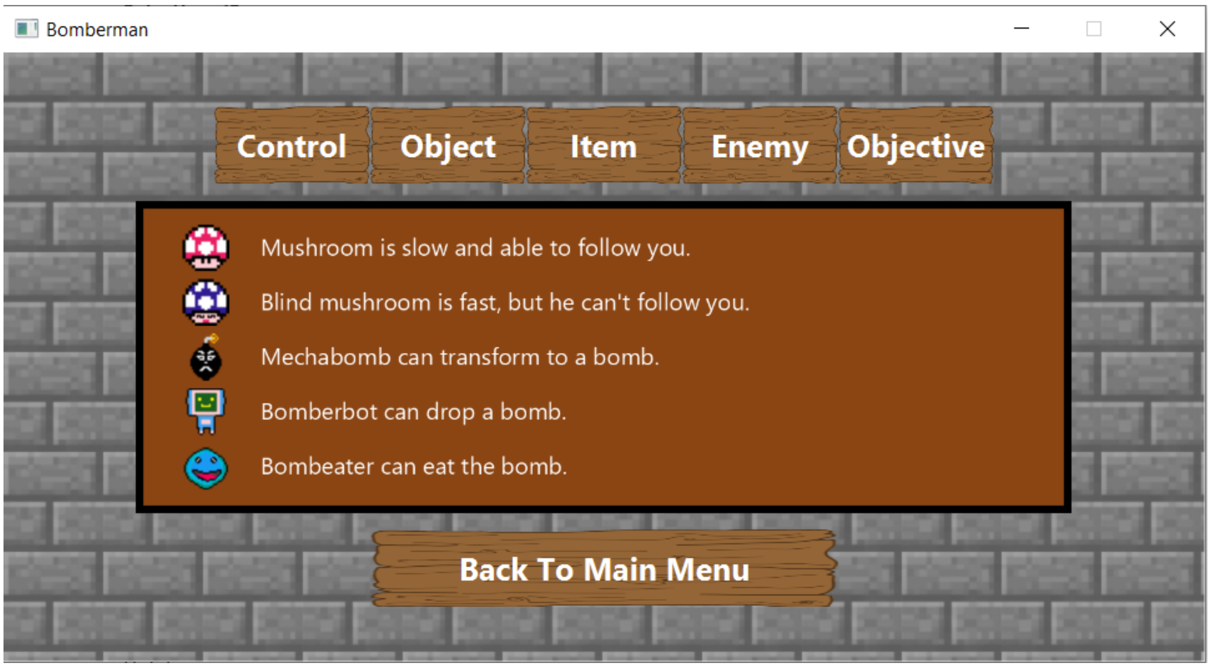
Start Game button will take you to a scene where you can select a level.

Shop button will take you to the shop.

Help button will tell you about how to play this game and information of everything in the game.

Quit button will quit the game.

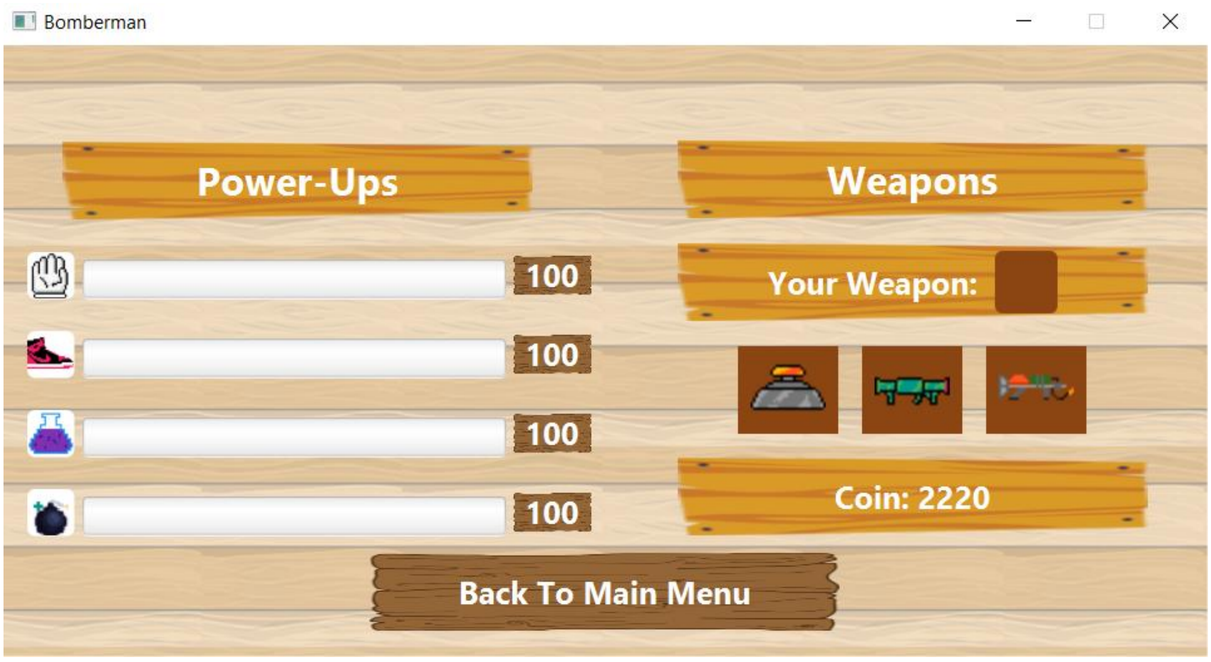
HelpScene



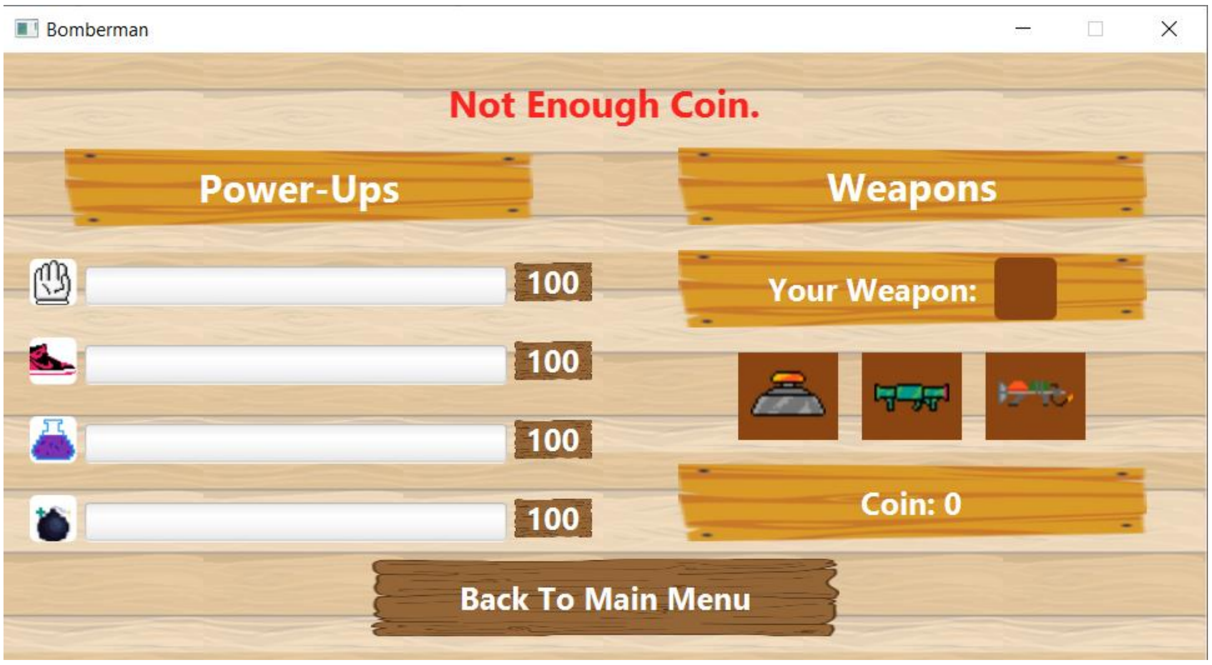
Mushroom is slow and able to follow you.
Blind mushroom is fast, but he can't follow you.
Mechabomb can transform to a bomb.
Bomberbot can drop a bomb.
Bombeater can eat the bomb.

This scene will tell you about the information of this game. You can select by clicking the buttons at the top. For example, if you want to know about enemies, you can click enemy button and it will show the information about the enemies.

ShopScene

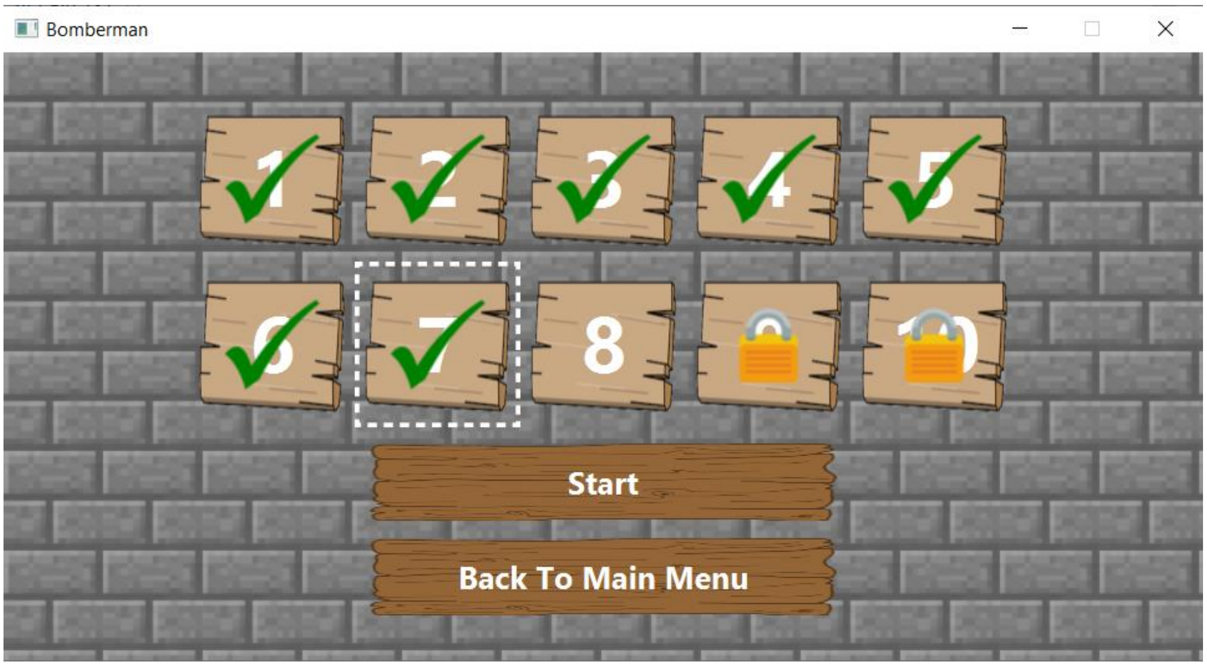


You can buy power-ups and weapons in this scene and it will tell you how much coins you have.

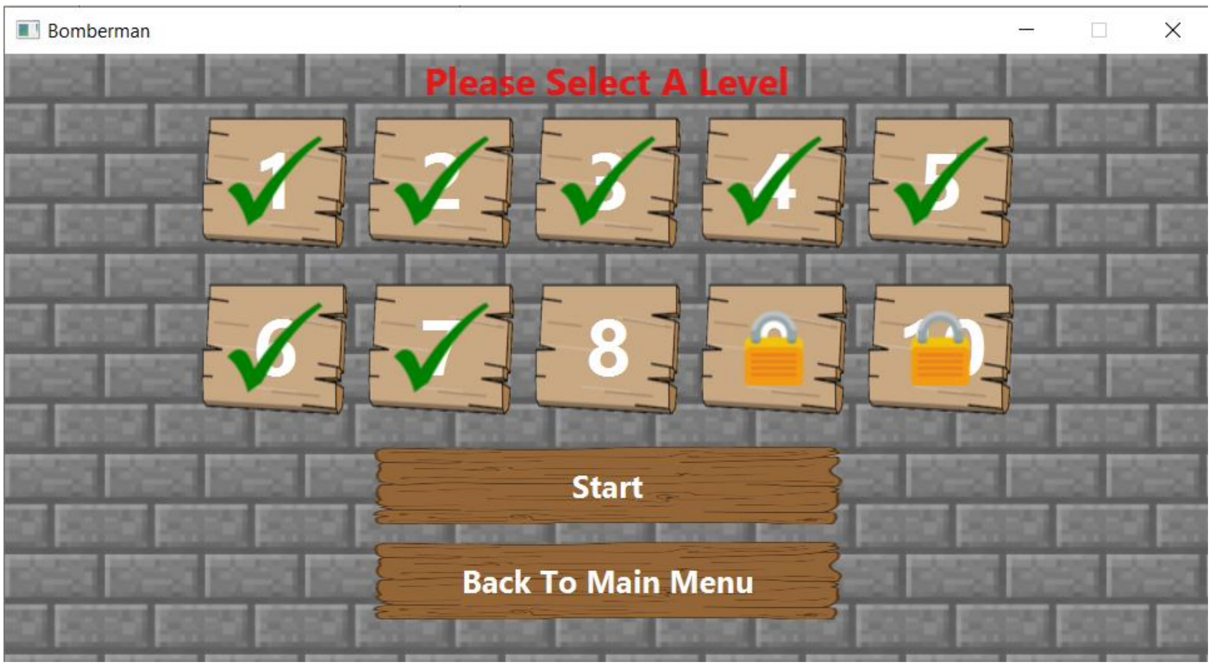


If you have not enough coin, the red message will show on the top of the screen.

SelectLevelScene

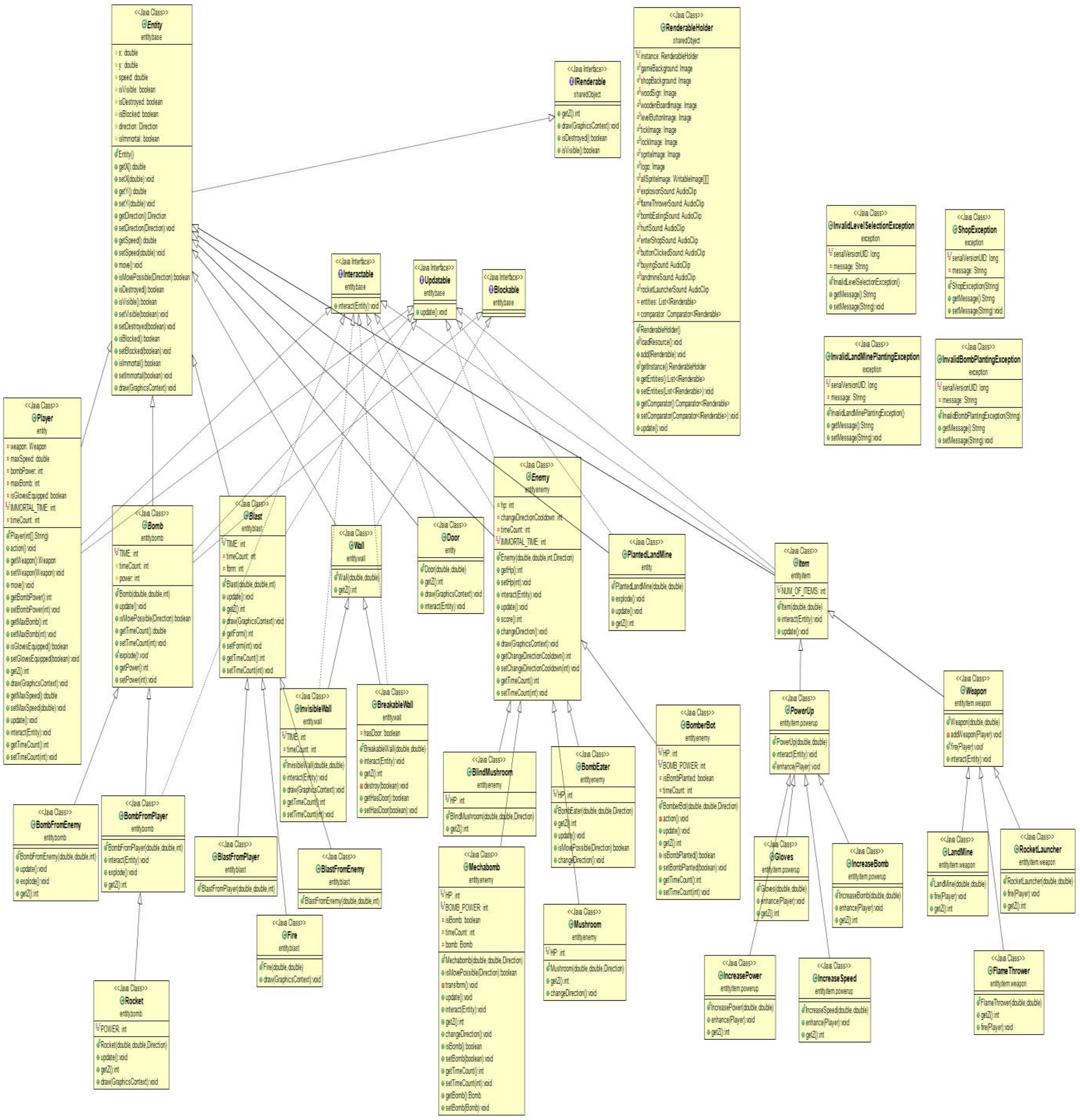


This scene will tell you which level you have completed, and which level is locked. You must select a level before you click start. There will be a white frame on a selected level.

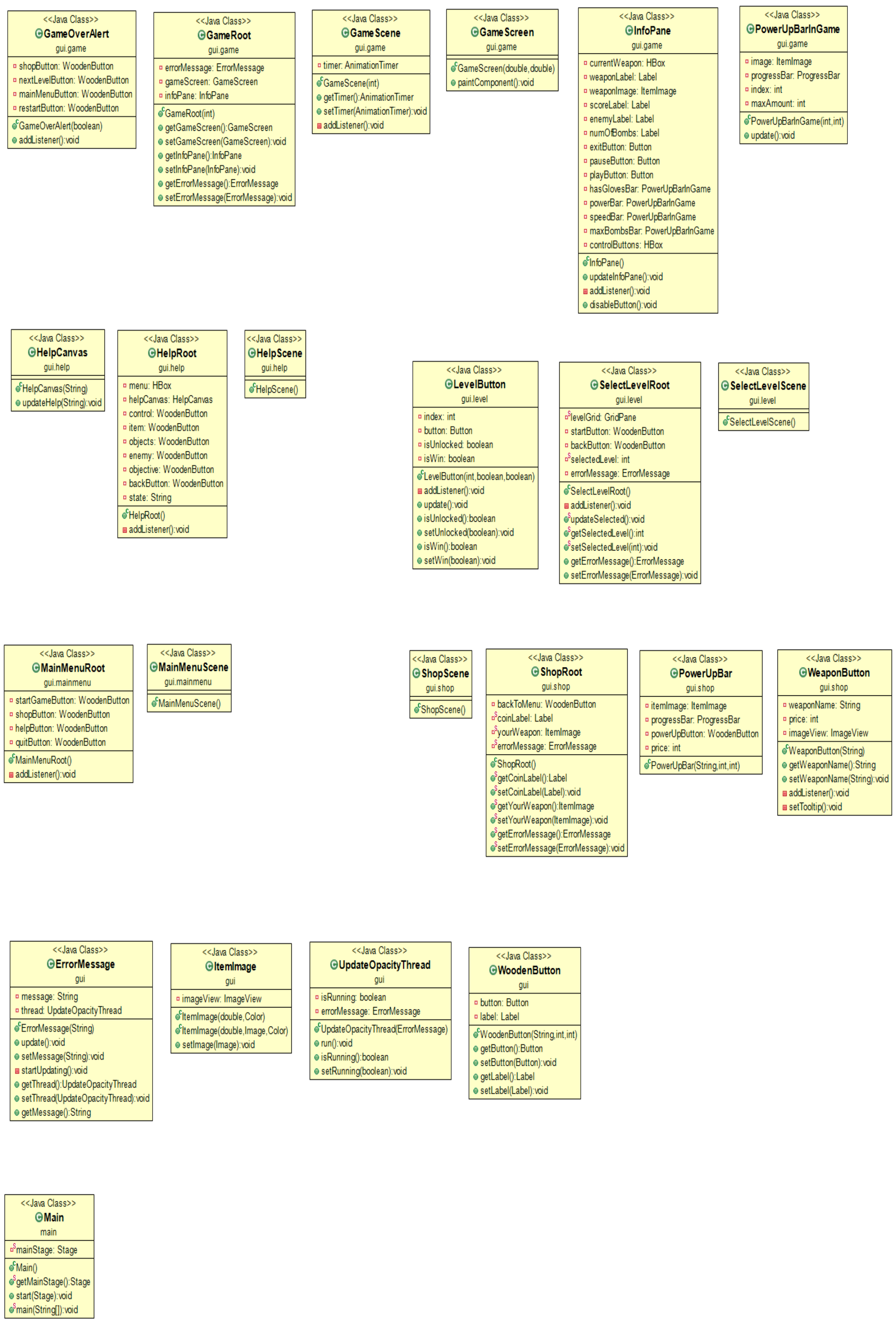


If you don’t select a level before you click start, there will be red message on the top of the screen.

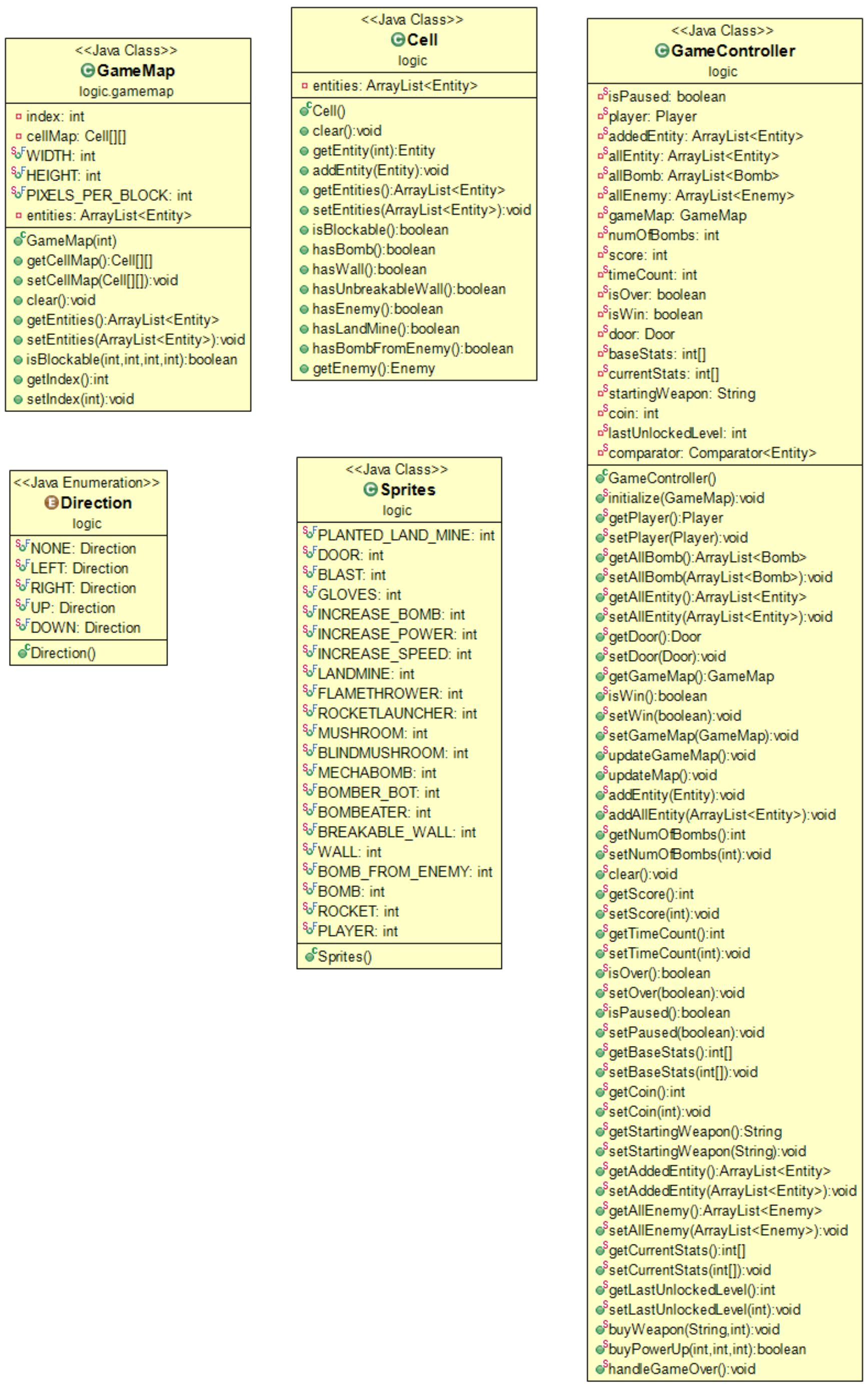
Class Diagram



Class Diagram



Class Diagram



1. Package entity

1.1. Class Door extends Entity implements Interactable

1.1.1. Constructor

+ Door(double x, double y)	Initialize fields

1.1.2. Methods

+ getZ()	Return the value Sprites.DOOR
+ draw(GraphicsContext gc)	Draw the sprite using the image in RenderableHolder. If the number of enemies equals 0, draw the open door. Otherwise, draw the closed door.
+ interact(Entity e)	If e is Player and the number of enemies in the map is 0, set isWin and isOver in GameController to true.

1.2 Class PlantedLandMine extends Entity implements Updatable

1.2.1. Constructor

+ Door(double x, double y)	Initialize fields

1.2.2. Methods

+ getZ()	Return the value Sprites.PLANTED_LAND_MINE
+ explode()	Create BlastFromPlayer at this cell and play the sound from RenderableHolder.
+ update()	If enemy is in the same cell, this will explode and set isDestroyed to true.

1.3 Class Player extends Entity implements Interactable, Updatable

1.3.1 Fields

- Weapon weapon	Weapon that the player has.
- double maxSpeed	Maximum speed that player can walk.
- int bombPower	Power of bombs that player place.
- int maxBomb	Maximum number of bomb that player can place.
- boolean isGlovesEquipped	A boolean that check if the player has gloves.
- int timeCount	A variable that increments by 1 every frame.
- static final IMMORTAL_TIME	The amount of time that player is immortal at the beginning of the game.

1.3.2 Constructor

+ Player(int[] baseStats, String startingWeapon)	Initialize fields using baseStats and startingWeapon.

1.3.3 Methods

+ void action() throws InvalidBombPlantingException, InvalidLandMinePlantingException	If weapon is null, generate a bomb at the cell on which player is standing. Throw InvalidBombPlantingException if there is a bomb, door, landmine on that cell or the number of bomb is exceeded. If weapon is not null, use the weapon by calling fire. Throw InvalidLandMinePlantingException if there already is a Planted land mine on that cell.
+ void move()	If a move is possible, call move method of parent class . If there is a interactable object on that cell, call interact method of that object.
+ int getZ()	Return the value Sprites.PLAYER
+ void update() throws InvalidBombPlantingException, InvalidLandMinePlantingException	Increment timeCount by 1. If timeCount equals IMMORTAL_TIME, set player's isImmortal to false. Call move() and change direction by using InputUtility.getKeyPressed(). If there is an Enemy e nearby, call this.interact(e). If a player is on the same cell as Door d, call d.interact(this).
+ void interact(Entity e)	If e is instance of Blast or Enemy, isImmortal is false and e is not instance of Fire, play a sound and set isOver to true.
+ void draw(GraphicsContext gc)	Draw a player using player's Images in RenderableHolder.
+ Weapon getWeapon()	Getter for weapon
+ void setWeapon(Weapon weapon)	Setter for weapon
+ int getBombPower()	Getter for bombPower
+ void setBombPower(int bombPower)	Setter for bombPower
+ int getMaxBomb()	Getter for maxBomb
+ void setMaxBomb(int maxBomb)	Setter for maxBomb
+ boolean isGlovesEquipped()	Getter for isGlovesEquipped
+ void setGlovesEquipped(boolean isGlovesEquipped)	Setter for isGlovesEquipped
+ double getMaxSpeed()	Getter for maxSpeed
+ void setMaxSpeed(double maxSpeed)	Setter for maxSpeed

2. Package entity.base

2.1 Abstract Class Entity implements IRenderable

2.1.1 Fields

# double x	Position of the object in the horizontal axis
# double y	Position of the object in the vertical axis
# double speed	Speed of the object
# boolean isVisible	Check if the object is visible
# boolean isDestroyed	Check if the object is destroyed
# boolean isBlocked	Check if the object is blocked by moving bombs
# Direction direction	Direction of the object
# boolean isImmortal	Check if the object can't be destroyed

2.1.2 Constructor

+ Entity()	Initialize fields

2.1.3 Methods

+ boolean isMovePossible(Direction d)	Check if a move in Direction d is possible
+ void move()	Move the entity in its direction by its speed
+ double getX()	Getter for x
+ void setX(double x)	Setter for x
+ double getY()	Getter for y
+ void setY(double y)	Setter for y
+ Direction getDirection()	Getter for direction
+ void setDirection(Direction direction)	Setter for direction
+ double getSpeed()	Getter for speed
+ void setSpeed(double speed)	Setter for speed
+ boolean isDestroyed()	Getter for isDestroyed
+ void setDestroyed(boolean isDestroyed)	Setter for isDestroyed
+ boolean isVisible()	Getter for isVisible
+ void setVisible(boolean isVisible)	Setter for isVisible
+ boolean isBlocked()	Getter for isBlocked
+ void setBlocked(boolean isBlocked)	Setter for isBlocked
+ boolean isImmortal()	Getter for isImmortal
+ void setImmortal(boolean isImmortal)	Setter for isImmortal
+ void draw(GraphicsContext gc)	Draw an image which is in RenderableHolder at (x,y)

2.2 Interface Blockable (This interface has no method.)

2.3 Interface Updatable

2.3.1 Method

+ abstract void update()	This method is call every frame of the game.

2.4 Interface Interactable

2.4.1 Method

+ abstract void interact(Entity e)	This method is called when two object is in the same cell or the first object runs into the second object.

3. Package entity.blast

3.1 Abstract Class Blast extends Entity Implements Updatable

3.1.1 Fields

+ static final int TIME	The amount of time that a blast will be on the cell.
- int timeCount	This variable will decrement by 1 every time method update() is called.
- int form	This variable represent the form of the blast used in draw(). 0 is center, 1 is horizontal, 2 is vertical.

3.1.2 Constructor

+ Blast(double x, double y, int form)	Initialize fields

3.1.3 Methods

+ void update()	Call interact() method of every interactable object on the same cell as this blast. If timeCount < 1, destroy this object.
+ int getZ()	Return Sprites.BLAST
+ void draw(GraphicsContext gc)	Draw an blast's image in RenderableHolder
+ int getForm()	Getter for form
+ void setForm(int form)	Setter for form
+ int getTimeCount()	Getter for timeCount
+ void setTimeCount(int timeCount)	Setter for timeCount

3.2 Class BlastFromPlayer extends Blast

3.2.1 Constructor

+ BlastFromPlayer(double x, double y, int form)	Initialize fields

3.3 Class BlastFromEnemy extends Blast

3.3.1 Constructor

+ BlastFromEnemy(double x, double y, int form)	Initialize fields

3.3 Class FIre extends Blast

3.3.1 Constructor

+ Fire(double x, double y)	Initialize fields

3.3.2 Method

+ draw(GraphicsContext gc)	Draw fire image which is in RenderableHolder

4. Package entity.bomb

4.1 Abstract Class Bomb implements Updatable, Blockable

4.1.1 Fields

+ static final int TIME	The amount of time before the bomb explode
# int timeCount	This variable will decrement every time update() is called.
# int power	The length of blast after the bomb explode

4.1.2 Constructor

+ Bomb(double x, double y, int power)	Initialize fields

4.1.3 Methods

+ void update()	- timeCount will decrease by 1. - If timeCount < 1, the bomb will explode. - call move() method
+ boolean isMovePossible(Direction d)	- Check if a move in Direction d is possible - Deal with the case that the bomb collide with enemy
+ int getTimeCount()	Getter for timeCount
+ void setTimeCount(int timeCount)	Setter for timeCount
+ int getPower()	Getter for power
+ void setPower(int power)	Setter for power
+ abstract void explode()	This method is called when timeCount is less than 1.

4.2 Class BombFromPlayer extends Bomb implements Interactable

4.2.1 Constructor

+ BombFromPlayer(double x, double y, int power)	Initialize fields

4.2.2 Methods

+ void interact(Entity e)	- If e is instance of Blast, set timeCount to 0 - If e is instance of Player and e has gloves, set direction of this to be the same direction as e and set speed to 5. - If e is instance of BombEater, destroy this.
+ void explode()	Generate the blasts in 4 direction with the length of bomb's power.
+ int getZ()	Return Sprites.BOMB

4.3 Class Rocket extends BombFromPlayer

4.3.1 Field

+ static final int POWER	Power of the rocket

4.3.2 Constructor

+ Rocket(double x, double y, Direction d)	Initialize fields

4.3.3 Methods

+ void update()	- move and decrease timeCount by 1 - if timeCount < 1, it will explode. - if this object collides with enemy, it will explode.
+ int getZ()	Return Sprites.ROCKET
+ void draw(GraphicsContext gc)	Draw fire image which is in RenderableHolder

4.4 Class BombFromEnemy extends Bomb

4.4.1 Constructor

+ BombFromEnemy(double x, double y, int power)	Initialize fields

4.4.2 Methods

+ void update()	- decrease timeCount by 1 - if timeCount < 1, it will explode.
+ void explode()	Generate the blasts in 4 direction with the length of bomb's power.
+ int getZ()	Return Sprites.BOMB_FROM_ENEMY

5. Package Enemy

5.1 Abstract Class Enemy implements Updatable, Interactable

5.1.1 Fields

- int hp	Hp of the enemy
- int changeDirectionCooldown	Amount of time that enemy cannot change direction
# int timeCount	A variable that increments by 1 every time update() is called.
- static final IMMORTAL_TIME	Amount of time that enemy is immortal after interact with blast.

5.1.2 Constructor

+ Enemy(double x, double y, int hp, Direction d)	Initialize fields

5.1.3 Methods

+ void interact(Entity e)	If e is instance of BlastFromPlayer or Fire, decrease enemy's hp by 1 and set isImmortal to true.
+ void update()	- If isImmortal is true, increase timeCount by 1. - If timeCount equals IMMORTAL_TIME, set isImmortal to false and set timeCount to 0. - Move and change direction
+ int score()	Return 50 * (this.getZ() - Sprites.MUSHROOM + 1)
+ void changeDirection()	Randomly change direction.
+ int getHp()	Getter for hp
+ void setHp(int hp)	Setter for hp
+ void draw(GraphicsContext gc)	Draw the image which is in RenderableHolder. If isImmortal is true, the image will flicker.
+ int getChangeDirectionCooldown()	Getter for changeDirectionCooldown
+ void setChangeDirectionCooldown(int changeDirectionCooldown)	Setter for changeDirectionCooldown
+ int getTimeCount()	Getter for timeCount
+ void setTimeCount(int timeCount)	Setter for timeCount

5.2 Class Mushroom extends Enemy

5.2.1 Fields

+ static final int HP	Hp of Mushroom

5.2.2 Constructor

+ Mushroom(double x, double y, Direction d)	Initialize fields

5.2.3 Methods

+ int getZ()	Return Sprites.MUSHROOM
+ void changeDirection()	If player is near(3 cells away), follow the player. Otherwise, randomly change direction

5.3 Class BlindMushroom extends Enemy

5.3.1 Fields

+ static final int HP	Hp of BlindMushroom

5.3.2 Constructor

+ BlindMushroom(double x, double y, Direction d)	Initialize fields

5.3.3 Methods

+ int getZ()	Return Sprite.BLINDMUSHROOM

5.4 Class Mechabomb extends Enemy

5.4.1 Fields

+ static final int HP	Hp is Mechabomb
+ static final int BOMB_POWER	Power of bomb that mechabomb transform to
- boolean isBomb	Boolean that check if mechabomb is being a bomb
- Bomb bomb	A bomb that mechabomb tranform to

5.4.2 Constructor

+ Mechabomb (double x, double y, Direction d)	Initialize fields

5.4.3 Methods

+ boolean isMovePossible(Direction d)	If isBomb is true, return false. Otherwise call parent method.
- void tranform()	- Generate a bomb at the cell and set isVisible of mechabomb to false. - Set speed to 0 - Set isBomb to true
+ void update()	- Check if it's time to transform back to mechabomb and set isBomb to false. - Randomly transform to bomb
+ void interact(Entity e)	If isBomb is true, it cannot interact with other entity.

+ int getZ()	Return Sprites.MECHABOMB
+ void ChangeDirection()	If player is near(3 cells away), follow the player. Otherwise, randomly change direction
+ boolean isBomb()	Getter for isBomb
+ void setBomb(boolean isBomb)	Setter for isBomb
+ Bomb getBomb()	Getter for bomb
+ void setBomb(Bomb bomb)	Setter for bome

5.5 Class BomberBot extends Enemy

5.5.1 Fields

+ static final int HP	Hp of BomberBot
+ static final int BOMB_POWER	Power of a bomb that BomberBot place
- boolean isBombPlanted	Boolean that checks if a bomb is placed
- static final int COOLDOWN	The period in which BomberBot can't place a bomb

5.5.2 Constructor

+ BomberBot(double x, double y, Direction d)	Initialize fields

5.5.3 Methods

- void action()	Generate the bomb at the same cell and set isBombPlanted to true.
+ void update()	-If there is no bomb that this BomberBot place, set isBombPlanted to false. -If isBombPlanted is false, randomly call action().
+ int getZ()	Return Sprites.BOMBER_BOT
+ boolean isBombPlanted()	Getter for isBombPlanted
+ void setBombPlanted(boolean isBombPlanted)	Setter for isBombPlanted

5.6 Class BombEater extends Enemy

5.6.1 Fields

+ static final int HP	Hp of BombEater

5.6.2 Constructor

+ BombEater(double x, double y, Direction d)	Initialize fields

5.6.3 Methods

+ int getZ()	Return Sprite.BOMBEATER

+ void update()	Call interact to every bomb on the same cell as this BombEater.
+ boolean isMovePossible(Direction d)	- Check if a move in Direction d is possible - BombEater can move pass BombFromPlayer
+ void changeDirection()	If the bomb is near(3 cells away), follow the bomb. Otherwise, randomly change direction

6. Package entity.item

6.1 Abstract Class Item extends Entity implements Updatable, Interactable

6.1.1 Fields

+ static final int NUM_OF_ITEMS	Number of items in the game

6.1.2 Constructor

+ Item(double x, double y)	Initialize fields

6.1.3 Methods

+ void interact(Entity e)	If e is instance of Blast, destroy this.
+ void update()	If Player p is on the same cell as this item, call interact(p).

7. Package entity.powerup

7.1 Abstract Class PowerUp extends Item

7.1.1 Constructor

+ PowerUp(double x, double y)	Initialize fields

7.1.2 Methods

+ void interact(Entity e)	Enhance player’s stats
+ abstract void enhance(Player p)	This method is called every time this object interact with player.

7.2 Class Gloves extends PowerUp

7.2.1 Constructor

+ Gloves(double x, double y)	Initialize fields

7.2.2 Methods

+ int getZ()	Return Sprites.GLOVES

+ void enhance(Player p)	Call p.setGlovesEquipped(true) and update GameController's currentStats

7.3 Class IncreasePower extends PowerUp

7.3.1 Constructor

+ IncreasePower(double x, double y)	Initialize fields

7.3.2 Methods

+ int getZ()	Return Sprite.INCRESES_POWER
+ void enhance(Player p)	Increase p's bombPower by 1 and update GameController's currentStats

7.4 Class IncreaseBomb extends PowerUp

7.4.1 Constructor

+ IncreaseBomb (double x, double y)	Initialize fields

7.4.2 Methods

+ int getZ()	Return Sprites.INCREASE_BOMB
+ void enhance(Player p)	Increase p's maxBomb by 1 and update GameController's currentStats

7.5 Class IncreaseSpeed extends PowerUp

7.5.1 Constructor

+ IncreaseSpeed(double x, double y)	Initialize fields

7.5.2 Methods

+ int getZ()	Return Sprites.INCREASE_SPEED
+ void enhance(Player p)	Increase p's maxSpeed by 0.1 and update GameController's currentStats

8. Package entity.item.weapon

8.1 Abstract Class Weapon extends Item

8.1.1 Constructor

+ Weapon(double x, double y)	Initialize fields

8.1.2 Methods

- void addWeapon(Player p)	Set weapon of p to this weapon

+ abstract void fire(Player p) throws InvalidLandMinePlantingException	This method is called every time p.action() is called and p has a weapon.
+ void interact(Entity e)	Add this weapon to p

8.2 Class LandMine extends Weapon

8.2.1 Constructor

+ LandMine(double x, double y)	Initialize fields

8.2.2 Methods

+ void fire(Player p) throws InvalidLandMinePlantingException	Generate PlantedLandMine at the same cell as p. If there already is PlantedLandMine on that cell, throw InvalidLandMinePlantingException
+ int getZ()	Return Sprite.LANDMINE

8.3 Class RocketLauncher extends Weapon

8.3.1 Constructor

+ RocketLauncher (double x, double y)	Initialize fields

8.3.2 Methods

+ void fire(Player p)	Launch a Rocket with the same direction as the player and the speed of 5
+ int getZ()	Return Sprite.ROCKET_LAUNCHER

8.4 Class FlameThrower extends Weapon

8.4.1 Constructor

+ FlameThrower(double x, double y)	Initialize fields

8.4.2 Methods

+ void fire(Player p)	Create 3 Fires in front of the player
+ int getZ()	Return Sprite.FLAMETHROWER

9. Package entity.wall

9.1 Class Wall implements Blockable

9.1.1 Constructor

+ Wall(double x, double y)	Initialize fields

9.1.2 Methods

+ int getZ()	Return Sprite.WALL

9.2 Class BreakableWall extends Wall implement Interactable

9.2.1 Fields

- boolean hasDoor	Boolean that checks if there is Door behind the wall

9.2.2 Constructor

+ BreakableWall(double x, double y)	Initialize fields

9.2.3 Methods

+ void interact(Entity e)	If e is instance of BlastFromPlayer, destroy this and call this.destroy(true)
+ int getZ()	Return Sprites.BREAKABLE_WALL
- destroy(boolean isDropped)	- If hasDoor is true, generate Door at that cell. Otherwise, randomly generate Item at that cell - If isDropped is false, Item will not be generated.
+ boolean getHasDoor()	Getter for hasDoor
+ void setHasDoor()	Setter for hasDoor

9.3 Class InvisibleWall extends Wall implement Interactable

9.3.1 Fields

+ static final int TIME	Amount of time until the wall disappear
- int timeCount	A variable that increments by 1 every frame

9.3.2 Constructor

+ InvisibleWall(double x, double y)	Initialize fields

9.3.3 Methods

+ void interact(Entity e)	Set isVisible to true and set timeCount to 0
+ draw(GraphicsContext gc)	- Draw the image which is in RenderableHolder. - Change opacity every frame.
+ int getTimeCount()	Getter timeCount
+ void setTImeCount(int timeCount)	Setter timeCount

10. Package exception

10.1 Class InvalidBombPlantingException extends Exception

10.1.1 Field

- String message	Message of the exception

10.1.2 Constructor

+ InvalidBombPlantingException(String message)	Initialize fields

10.1.3 Methods

+ String getMessage()	Getter for message
+ void setMessage(String message)	Setter for message

10.2 Class InvalidLandMinePlantingException extends Exception

10.2.1 Field

- String message	Message of the exception

10.2.2 Constructor

+ InvalidLandMinePlantingException()	Initialize fields

10.2.3 Methods

+ String getMessage()	Getter for message
+ void setMessage(String message)	Setter for message

10.3 Class InvalidLevelSelectionException extends Exception

10.3.1 Field

- String message	Message of the exception

10.3.2 Constructor

+ InvalidLevelSelectionException()	Initialize fields

10.3.3 Methods

+ String getMessage()	Getter for message
+ void setMessage(String message)	Setter for message

10.4 Class ShopException extends Exception

10.4.1 Field

- String message	Message of the exception

10.4.2 Constructor

+ ShopException(String message)	Initialize fields

10.4.3 Methods

+ String getMessage()	Getter for message
+ void setMessage(String message)	Setter for message

11. Package gui

11.1 Class ErrorMessage extends Label

11.1.1 Field

- String message	Message of the label
- UpdateOpacityThread thread	Thread that change opacity of the message

11.1.2 Constructor

+ ErrorMessage(String message)	Initialize fields

11.1.3 Methods

+ void update()	Decrease opacity of the label by 0.01
+ void setMessage(String message)	Setter for message and call startUpdating()
+ String getMessage()	Getter for message
+ void setThread(UpdateOpacityThread thread)	Setter for thread
+ UpdateOpacityThread getThread()	Getter for thread
- void startUpdating()	Start thread

11.2 Class ItemImage extends StackPane

11.2.1 Fields

- ImageView imageView	ImageView of the item

11.2.2 Constructor

+ ItemImage(double width, Color backgroundColor)	Initialize imageView as empty Imageview
+ ItemImage(double width, Image image, Color backgroundColor)	Initialize imageView

11.2.3 Methods

+ void setImage(Image image)	Set Image of ImageView

11.3 Class UpdateOpacityThread extends Thread

11.3.1 Fields

- boolean isRunning	Boolean that checks if the thread is running
- ErrorMessage errorMessage	ErrorMessage that has this thread as its field.

11.3.2 Constructor

+ UpdateopacityThread(ErrorMessage errorMessage)	Initialize fields

11.3.3 Method

+ void run()	Call update errorMessage every 15 ms
+ boolean isRunning()	Getter for isRunning
+ void setRunning(boolean isRunning)	Setter for isRunning

11.4 Class WoodenButton extends StackPane

11.4.1 Fields

- Button button	Button component of the object
- Label label	Label on the object

11.4.2 Constructor

+ WoodenButton(String text, int width, int height)	Initialize fields

11.4.3 Methods

+ Button getButton()	Getter for button
+ void setButton(Button button)	Setter for button
+ Label getLabel()	Getter for label
+ void setLabel(Label label)	Setter for label

12. Package gui.game

12.1 Class GameOverAlert extends Vbox

12.1.1 Fields

- WoodenButton shopButton	WoodenButton that change scene to ShopScene
- WoodenButton nextLevelButton	WoodenButton that change scene to GameScene in the next level

- WoodenButton mainMenuButton	WoodenButton that change scene to MainMenuScene
- WoodenButton restartButton	WoodenButton that change scene to GameScene of this level

12.1.2 Constructor

+ GameOverAlert(boolean isWin)	Initialize fields

12.1.3 Methods

+ void addListener()	Add listener to all WoodenButton

12.2 Class GameRoot extends StackPane

12.2.1 Fields

- ErrorMessage errorMessage	ErrorMessage that appear at the top of the scene.
- GameScreen gameScreen	GameScreen component of the root
- InfoPane infoPane	Pane that contains information of the game

12.2.2 Constructor

+ GameRoot(int index) throws InvalidLevelSelectionException	Initialize fields and throw InvalidLevelSelectionException if index is invalid.

12.2.3 Methods

+ GameScreen getGameScreen()	Getter for gameScreen
+ void setGameScreen(GameScreen gameScreen)	Setter for gameScreen
+ InfoPane getInfoPane()	Getter for infoPane
+ void setInfoPane(InfoPane infoPane)	Setter for infoPane
+ ErrorMessage getErrorMessage()	Getter for errorMessage
+ void setErrorMessage(ErrorMessage errorMessage)	Setter for errorMessage

12.3 Class GameScene extends Scene

12.3.1 Fields

- AnimationTimer timer	AnimationTimer that start when initialize GameScene

12.3.2 Constructor

+ GameScene(int index)	Initialize scene with GameRoot as a root and start timer

12.3.3 Methods

+ AnimationTimer getTimer()	Getter for timer
+ void setTimer(AnimationTimer timer)	Setter for timer
- void addListener()	Receive input from keyboard add it to InputUtility

12.4 Class GameScreen extends Canvas

12.4.1 Constructor

+ GameScreen(double width, double height)	Initialize canvas with given width and height

12.4.2 Methods

+ void paintComponent()	Draw all component in RenderableHolder that is visible and not destroyed

12.5 Class InfoPane extends Vbox

12.5.1 Fields

- Hbox currentWeapon	Hbox that contains weaponLabel and weaponImage
- Label weaponLabel	Label with the text “Weapon: ”
- ItemImage weaponImage	Show weapon that player is using
- Label scoreLabel	Show score that player get
- Label enemyLabel	Show number of enemy
- Label numOfBombs	Show number of bombs and maximum number of bombs
- Button exitButton	Button that change scene to MainMenuScene
- Button pauseButton	Button that stop the timer
- Button playButton	Button that start the timer
- PowerUpBarInGame hasGlovesBar	Bar that show current stats of player (Gloves)
- PowerUpBarInGame powerBar	Bar that show current stats of player (Power)
- PowerUpBarInGame speedBar	Bar that show current stats of player (Speed)
- PowerUpBarInGame maxBombsBar	Bar that show current stats of player (Bombs)
- Hbox controlButtons;	Hbox that contain exitButton, pauseButton, playButton

12.5.2 Constructor

+ InfoPane()	Initialize fields

12.5.3 Methods

+ void updateInfoPane()	Update all components of InfoPane
- void addListener()	Add listener for exitButton, pauseButton, playButton
+ void disableButton()	Disable all button on the InfoPane

12.6 Class PowerUpBarInGame extends Hbox

12.6.1 Fields

- ItemImage image	ItemImage of the item
- ProgressBar progressBar	ProgressBar that show the stat
- int index	Index of the stats
- int maxAmount;	Maximum amount of that stat

12.6.2 Constructor

+ PowerUpBarInGame(int index, int maxAmount)	Initialize fields

12.6.3 Methods

+ void update()	Set progress of progressBar

13. Package gui.help

13.1 Class HelpCanvas extends Canvas

13.1.1 Constructor

+ HelpCanvas(String s)	Initialize canvas

13.1.2 Methods

+ void updateHelp(String s)	Update the canvas - If s is “Control”, draw how to control - If s is “Enemy”, draw information of enemies - If s is “Item”, draw information of items - If s is “Objective”, draw objective of the game - If s is “Object”, draw information of objects

13.2 Class HelpRoot extends Vbox

13.2.1 Fields

- Hbox menu	HBox that contains all WoodenButton
- HelpCanvas halpCanvas	Canvas that show information
- WoodenButton control	Button that will show controlling of the game
- WoodenButton item	Button that will show information of items
- WoodenButton objects	Button that will show information of objects
- WoodenButton enemy	Button that will show information of enemies
- WoodenButton objective	Button that will show objective of the game

- WoodenButton backButton	Button that change the scene to MainMenuScene
- String state	String that contains state of the canvas

13.2.2 Constructor

+ HelpRoot()	Initialize fields

13.2.3 Methods

- void addListener()	Add listener to all WoodenButton

13.3 Class HelpScene extends Scene

13.3.1 Constructor

+ HelpScene()	Initialize the Scene with HelpRoot as a root

14. Package gui.level

14.1 Class LevelButton extends StackPane

14.1.1 Fields

- int index	Number that is showed on the button
- Button button	Button that set selectedLevel of SelectLevelRoot to index
- boolean isUnlocked	Boolean that check if the level is unlocked
- boolean isWin	Boolean that check if the level has been completed

14.1.2 Constructor

+ LevelButton(int index, boolean isUnlocked, boolean isWin)	Initialize fields

14.1.3 Methods

- void addListener()	Add listener to button
+ void update()	Change the style of this object if it is selected
+ boolean isUnlocked()	Getter for isUnlocked
+ void setUnlocked(boolean isUnlocked)	Setter for isUnlocked
+ boolean isWin()	Getter for isWin
+ void setWin(boolean isWin)	Setter for isWin

14.2 Class SelectlevelRoot extends StackPane

14.2.1 Fields

- static GridPane levelGrid	GridPane that contains LevelButtons
- static int selectedLevel	Variable that contains index of selected level
- WoodenButton startButton	Button that start the game
- WoodenButton backButton	Button that change scene to MainMenuScene
- ErrorMessage errorMessage	ErrorMessage that appear on the top

14.2.2 Constructor

+ SelectLevelRoot()	Initialize fields

14.2.3 Methods

- void addListener()	Add listener for startButton and backButton
+ static void updateSelected()	Update all LevelButton in levelGrid
+ static int getSelectedLevel()	Getter for selectedLevel
+ static void setSelectedLevel(int selectedLevel)	Setter for selectedLevel
+ ErrorMessage getErrorMessage()	Getter for errorMessage
+ void setErrorMessage(ErrorMessage errorMessage)	Setter for errorMessage

14.3 Class SelectLevelScene extends Scene

14.3.1 Constructor

+ SelectLevelScene()	Initialize scene with SelectLevelRoot as a root

15. Package gui.mainmenu

15.1 Class MainMenuRoot extends Hbox

15.1.1 Fields

- WoodenButton startGameButton	Change the scene to SelectLevelScene
- WoodenButton shopButton	Change the scene to ShopScene
- WoodenButton helpButton	Change the scene to HelpScene
- WoodenButton quitButton	Quit the game

15.1.2 Constructor

+ MainMenuRoot()	Initialize fields

15.1.3 Methods

- void addListener()	Add listener for all WoodenButton

15.2 Class MainMenuScene extends Scene

15.2.1 Constructor

+ MainMenuScene()	Initialize scene with MainMenuRoot as a root

16. Package gui.shop

16.1 PowerUpBar extends Hbox

16.1.1 Fields

- ItemImage itemImage	ItemImage of the item
- ProgressBar progressBar	ProgressBar that show base stat of the player
- WoodenButton powerUpButton	Button that will upgrade stat
- int price	Price of the upgrade

16.1.2 Constructor

+ PowerUpBar(String name, int index, int maxAmount)	Initialize fields and add listener to powerUpButton

16.2 Class WeaponButton extends Button

16.2.1 Fields

- String weaponName	Name of the weapon
- int price	Price of the weapon
- ImageView imageView	imageView of the weapon

16.2.2 Constructor

+ WeaponButton(String name)	Initialize fields

16.2.3 Methods

+ String getWeaponName()	Getter for weaponName
+ void setWeaponName(String weaponName)	Setter for weaponName
- void addListener()	Add listener to this button which call buyWeapon in GameController
- void setTooltip()	Add Tooltip that show weaponName and price

16.3 Class ShopRoot extends VBox

16.3.1 Fields

- WoodenButton backToMenu	Button that change the scene to MainMenuScene
- static Label coinLabel	Label that show number of coin the player have
- static ItemImage yourWeapon	ItemImage of yourWeapon
- static ErrorMessage errorMessage	ErrorMessage that show on the top of the scene

16.3.2 Constructor

+ ShopRoot()	Initialize fields

16.3.3 Methods

+ static Label getCoinLabel()	Getter for coinLabel
+ static void setCoinLabel(Label coinLabel)	Setter for coinLabel
+ static ItemImage getYourWeapon()	Getter for yourWeapon
+ static void setYourWeapon(ItemImage yourWeapon)	Setter for yourWeapon
+ static ErrorMessage getErrorMessage()	Getter for errorMessage
+ static void setErrorMessage(ErrorMessage errorMessage)	Setter for errorMessage

16.4 Class ShopScene extends Scene

16.4.1 Constructor

+ ShopScene()	Initialize scene with ShopRoot as a root

17. Package input

17.1 Class InputUtility

17.1.1 Fields

- static boolean isSpaceTriggered	Boolean that checks if SPACE is triggered
- static ArrayList<KeyCode> keyPressed	ArrayList that contains all key that is pressed

17.1.2 Methods

+ static void setKeyPressed(KeyCode keycode, boolean pressed)	If pressed is true, add keycode to keyPressed. If pressed is false, remove keycode from keyPressed.
+ static boolean getKeyPressed(KeyCode keyCode)	Return true if keyCode is in keyPressed. Otherwise, return false.
+ static boolean isSpaceTriggered()	Getter for isSpaceTriggered
+ static void setSpaceTriggered(boolean isSpaceTriggered)	Setter for isSpaceTriggered
+ static ArrayList<KeyCode> getKeyPressed()	Getter for keyPressed
+ static void setKeyPressed(ArrayList<KeyCode> keyPressed)	Setter for keyPressed
+ static void postUpdate()	Set isSpaceTriggered to false. If game is over or pause, clear keyPressed.

18. Package logic

18.1 Class Cell

18.1.1 Fields

- ArrayList<Entity> entities	All entities in this Cell

18.1.2 Constructor

+ Cell()	Initialize entities

18.1.3 Methods

+ void clear()	Clear entities
+ Entity getEntity(int index)	Get entity with the index in entities
+ void addEntity(Entity entity)	Add entity to entities
+ ArrayList<Entity> getEntities()	Getter for entities
+ void setEntities(ArrayList<Entity> entities)	Setter for entities
+ boolean isBlockable()	Return true only if there is Blockable in the Cell
+ boolean hasBomb()	Return true only if there is a Bomb in the Cell
+ boolean hasWall()	Return true only if there is a Wall in the Cell
+ boolean hasUnbreakableWall()	Return true only if there is an unbreakable wall in the Cell
+ boolean hasEnemy()	Return true only if there is an Enemy in the Cell
+ boolean hasLandMine()	Return true only if there is a PlantedLandMine in the Cell
+ boolean hasBombFromEnemy()	Return true only if there is a BombFromEnemy in the Cell
+ Enemy getEnemy()	Return the first Enemy of the entities

18.2 Enum Direction

This enum represents direction. It contains the following values: LEFT, RIGHT, UP, DOWN and NONE.

18.3 Class Sprites

This class contains the constant used for rendering and organizing.

18.4 Class GameController

18.4.1 Fields

- static boolean isPaused	Boolean that checks if the game is paused
- static Player player	Player in the game
- static ArrayList<Entity> addedEntity	ArrayList that contains all entity that is added to the game. (All entity added to the game must be here before be added to allEnity)
- static ArrayList<Entity> allEntity	ArrayList that contains all entity
- static ArrayList<Bomb> allBomb	ArrayList that contains all Bomb
- static ArrayList<Enemy> allEnemy	ArrayList that contains all Enemy
- static GameMap gameMap	Current GameMap
- static int numOfBombs	Number of bombs that player placed
- static int score	Total score that player get

- static int timeCount	This variable will increase by 1 every time updateMap is called.
- static boolean isOver	Boolean that checks if the game is over
- static boolean isWIn	Boolean that checks if the player win
- static Door door	Door in the gameMap
- static int[] baseStats	baseStats is used when initialize Player
- static int[] currentStats	currentStats is the stats in the game
- static String startingWeapon	startingWeapon is used when initialize Player
- static int coin	Number of coin that player have
- static int lastUnlockedLevel	Last level that is unlocked
- static Comparator<Entity> comparator	Comparator to sort for allEntity

18.4.2 Methods

+ static void initialize(GameMap gameMap)	Intitialize the game
+ static Player getPlayer()	Getter for player
+ static void setPlayer(Player player)	Setter for player
+ static ArrayList<Bomb> getAllBomb()	Getter for allBomb
+ static void setAllBomb(ArrayList<Bomb> allBomb)	Setter for allBomb
+ static ArrayList<Entity> getAllEntity()	Getter for allEntity
+ static void setAllEntity(ArrayList<Entity> allEntity)	Setter for allEntity
+ static Door getDoor()	Getter for door
+ static void setDoor(Door door)	Setter for door
+ static GameMap getGameMap()	Getter for gameMap
+ static void setGameMap(GameMap gameMap)	Setter for gameMap
+ static void setWin(boolean isWIn)	Setter for isWin
+ static boolean isWin()	Getter for isWin
+ static void updateGameMap()	Update each cell in gameMap
+ static void updateMap() throws InvalidBombPlantingException, InvalidLandMinePlantingException	- Update every updatable entity - Increase timeCount by 1 - setNumOfBombs to size of allBomb
+ static void addEntity(Entity entity)	Add entity to allEntity If entity is instance of Bomb, add it to allBomb. If entity is instance of Enemy, add it to allEnemy.
+ static void addAllEntity(ArrayList<Entity> entities)	Add all entity in entities to allEntity
+ static int getNumOfBombs()	Getter for numOfBombs
+ static void setNumOfBombs(int numOfBombs)	Setter for numOfBombs
+ static void clear()	Clear everything in the game
+ static int getScore()	Getter for score
+ static void setScore(int score)	Setter for score
+ static int getTimeCount()	Getter for timeCount
+ static void setTImeCount(int timeCount)	Setter for timeCount
+ static boolean isOver()	Getter for isOver
+ static void setOver(boolean isOver)	Setter for isOver

+ static boolean isPaused()	Getter for isPaused
+ static void setPaused(boolean isPaused)	Setter for isPaused
+ static int[] getBaseStats()	Getter for baseStats
+ static void setBaseStats(int[] baseStats)	Setter for baseStats
+ static int getCoin()	Getter for coin
+ static void setCoin(int coin)	Setter for coin
+ static String getStartingWeapon()	Getter for startingWeapon
+ static void setStartingWeapon(String startingWeapon)	Setter for startingWeapon
+ static ArrayList<Entity> getAddedEntity()	Getter for addedEntity
+ static void setAddedEntity(ArrayList<Entity> addedEntity)	Setter for addedEntity
+ static ArrayList<Enemy> getAllEnemy()	Getter for allEnemy
+ static void setAllEnemy(ArrayList<Enemy> allEnemy)	Setter for allEnemy
+ static int[] getCurrentStats	Getter for currentStats
+ static void setCurrentStats(int[] currentStats)	Setter for currentStats
+ static int getLastUnlockedLevel()	Getter for lastUnlockedLevel
+ static void setLastUnlockedLevel(int lastUnlockedLevel)	Setter for lastUnlockedLevel
+ static void buyWeapon(String weaponName, int price) throws ShopException	Set startingWeapon to weaponName and decrease coin by price. Throw ShopException if there is not enough coin.
+ static void butPowerUp(int index, int price, int maxAmount) throws ShopException	Increase baseStats and decrease coin by price. Throw ShopException if there is not enough coin.
+ static void handleGameOver()	- If the game is over and the player win, set startingWeapon to the weapon that player have and increase the coin.

19. Package logic.gamemap

19.1 Class CSVParser

19.1.1 Methods

+ static String[][] readCSV(String filename)	Read the csv file and return array of String

19.2 Class GameMap

19.2.1 Fields

- int index	Level
- Cell[][] cellMap	All Cell in the map
+ static final int WIDTH	Width of game board
+ static final int HEIGHT	Height of game board
+ static final int PIXELS_PER_BLOCK	Number of pixels per one block
- ArrayLIst<Entity> entities	All entities in the game

19.2.2 Constructor

+ GameMap(int index) throws InvalidLevelSelectionException	- Initialize field.
	- read csv file and add all entities in the file
	- throw InvalidLevelSelectionException if index = 0
	W = Wall
	BW = BreakableWall
	IW = InvisibleWall
	A = BomberBot
	B = Mechabomb
	C = Mushroom
	D = BlindMushroom
	E = BombEater
	1 = UP, 2 = DOWN, 3 = LEFT, 4 = RIGHT

19.2.3 Methods

+ Cell[][] getCellMap()	Getter for cellMap
+ void setCellMap(Cell[][])	Setster for cellMap
+ void clear()	Clear all cell in the cellMap
+ ArrayList<Entity> getEntities()	Getter for entities
+ void setEntities(ArrayList<Entity> entities)	Setter for entities
+ boolean isBlockable(int axis, int index, int start, int stop)	Return true if there is Blockable entity between start and stop. Otherwise, return false. (axis: 0 = X, 1 = Y)
+ int getIndex()	Getter for index
+ void setIndex(int)	Setter for index

20. Package main

20.1 Class Main extends Application

20.1.1 Fields

- static Stage mainStage	The stage of this Application

20.1.2 Methods

+ static Stage getMainStage()	Getter for mainStage
+ void start(Stage primaryStage)	Set mainStage to primaryStage and set Scene of primaryStage to MainMenuScene
+ static void main(String[] args)	Start the application

21. package sharedObject

21.1 Interface IRenderable

21.1.1 Methods

+ int getZ()	Return sprites number of the entity
+ void draw(GraphicsContext gc)	Draw the image in RenderableHolder
+ boolean isDestroyed()	Boolean that checks if the entity is remove from the game.

+ boolean isVisible()	Boolean that checks if the entity is visible.

21.2 Class RenderableHolder

This class contains the images and audios that used for rendering. This class also contain all entities which are instances of IRenderable.

21.2.1 Fields

- static final RenderableHolder instance	instance of singleton class
+ static Image gameBackground	Image for the background
+ static Image shopBackground	Image for the shop’s background
+ static Image woodSign	Image for the wood sign
+ static Image woodenBoardImage	Image of wooden board
+ static Image levelButtonImage	Image for the levelButton
+ static Image tickImage	Image for completed level
+ static Image lockImage	Image for locked level
+ static Image spriteImage	Image for all sprites
+ static Image logo	Image for logo
+ static WritableImage[][] allSpriteImage	Array that contains all sprites image
+ static Audio bombEatingSound	Audio for BombEater
+ static Audio explosionSound	Audio played when bomb explode
+ static Audio flameThrowerSound	Audio played when player use FlameThrower
+ static Audio hurtSound	Audio played when player lose
+ static Audio enterShopSound	Audio played when enter shop
+ static Audio buttonClickedSound	Audio played when click button
+ static Audio buyingSound	Audio played when buy an item
+ static Audio landmineSound	Audio played when plant land mine
+ static Audio rocketLauncherSound	Audio played when launch rocket
+ List<IRenderable> entities	All entities that is instance of IRenderable
- Comparator<IRenderable> comparator	Comparator used to sort entities

21.2.1 Method

+ static void loadResource	Load all image and audio used in this game
+ static void add(IRenderable entity)	Add entity to entities
+ static RenderableHolder getInstance()	Getter for instance
+ static setEntities(List<IRenderable> entities)	Setter for entities
+ List<IRenderable> getEntities()	Getter for entities
+ Comparator<IRenderable> getComparator()	Getter for camparator
+ void setComparator(Comparator<IRenderable> comparator)	Setter for comparator
+ update()	Remove entities which are destroyed