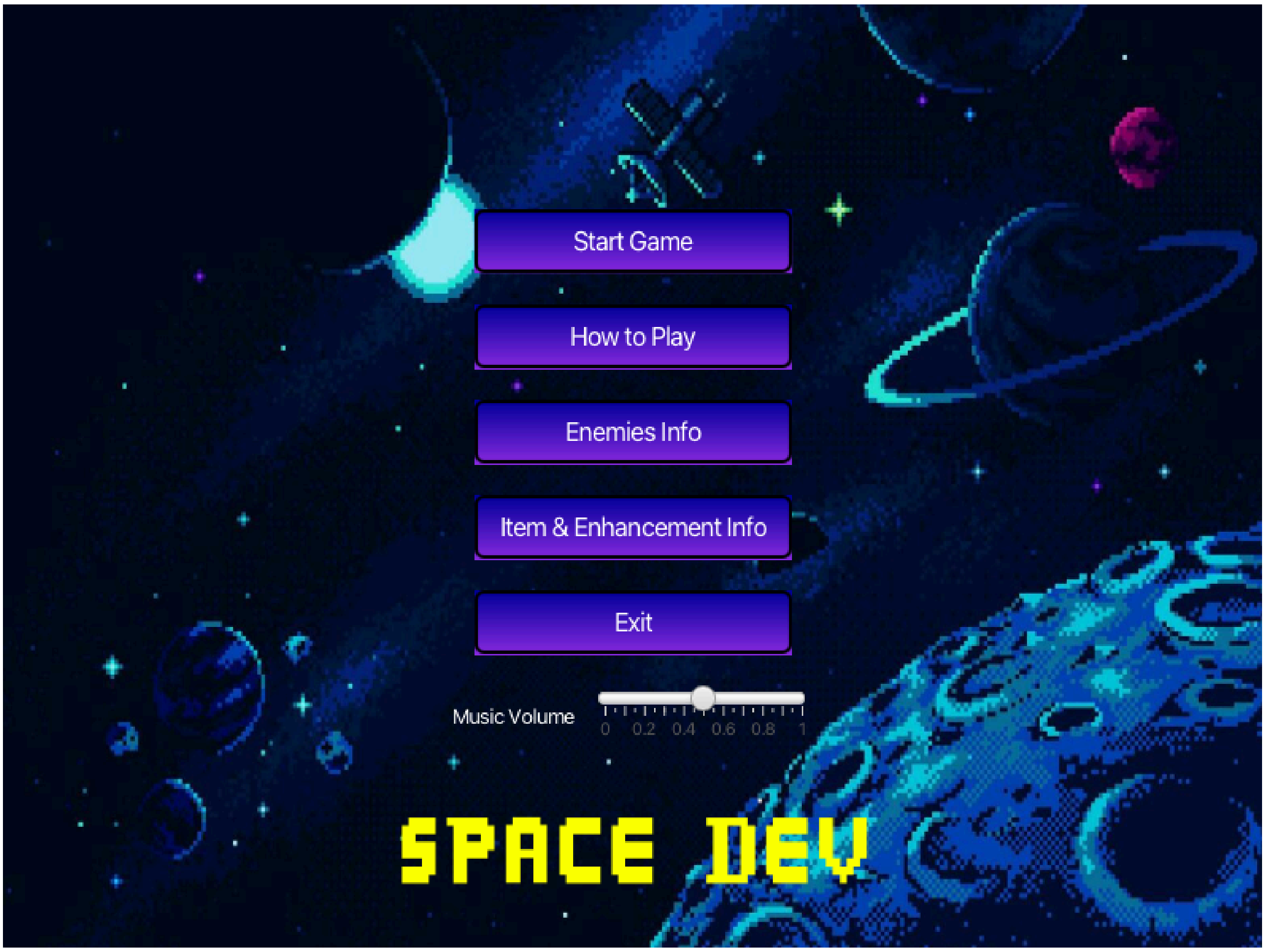
SpaceDev

Created by
6633241121 Sirasit Tanajirawat
6633233121 Sadit Wongprayon
6633138221 Papawin Tangchitporn
6633038121 Jaitnipat Wichitniti

2110215 Programming Methodology
Semester 2 Year 2023
Chulalongkorn University

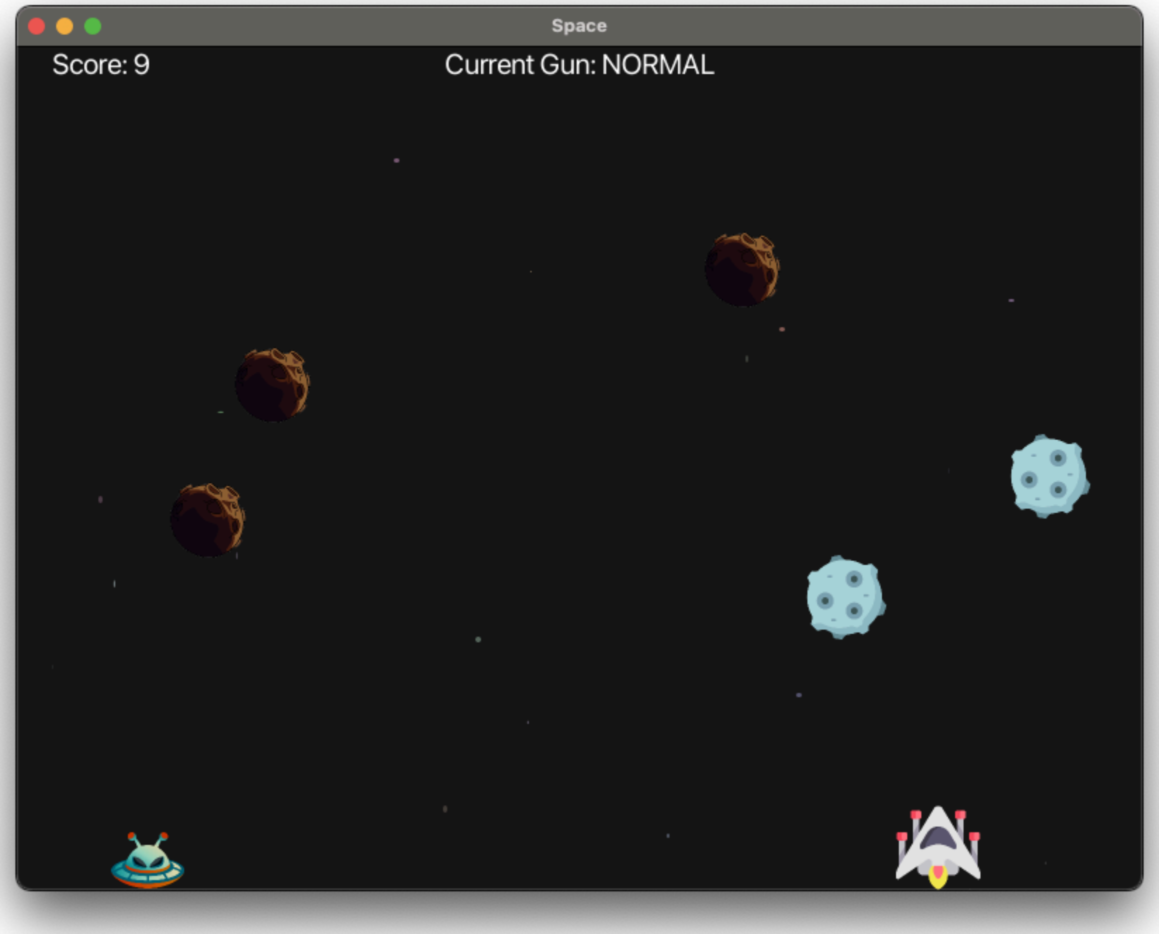
1. Introduction to the Game



Welcome to "Space Dev," a thrilling arcade-style space shooter game that puts you in control of a powerful rocket against waves of incoming enemy bombs. In this game, your objective is to survive as long as possible, dodging bombs and shooting them down to rack up points. Let's delve into the exciting features and mechanics that make this game an engaging and action-packed experience.

Gameplay Overview

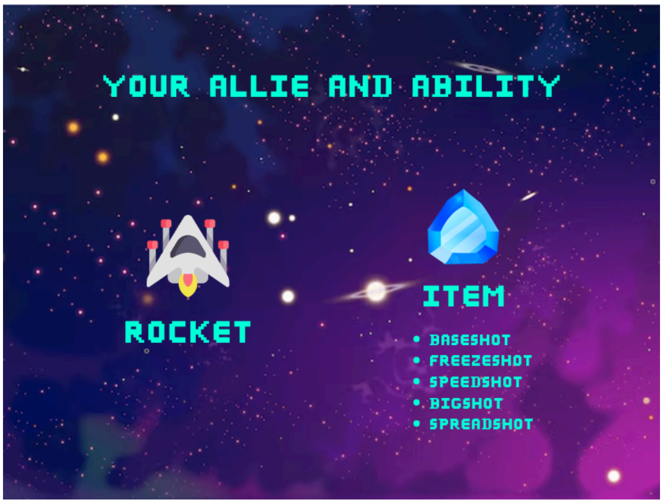
"Space Dev" offers fast-paced and dynamic gameplay. You control a rocket situated at the bottom of the screen, moving horizontally to avoid incoming bombs while shooting them down to defend yourself.

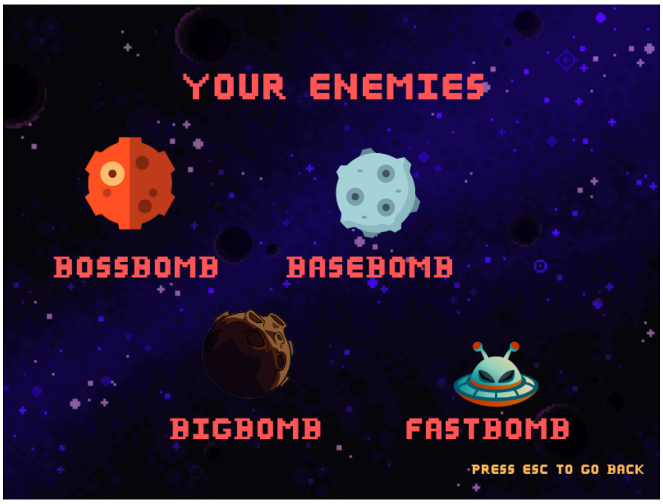


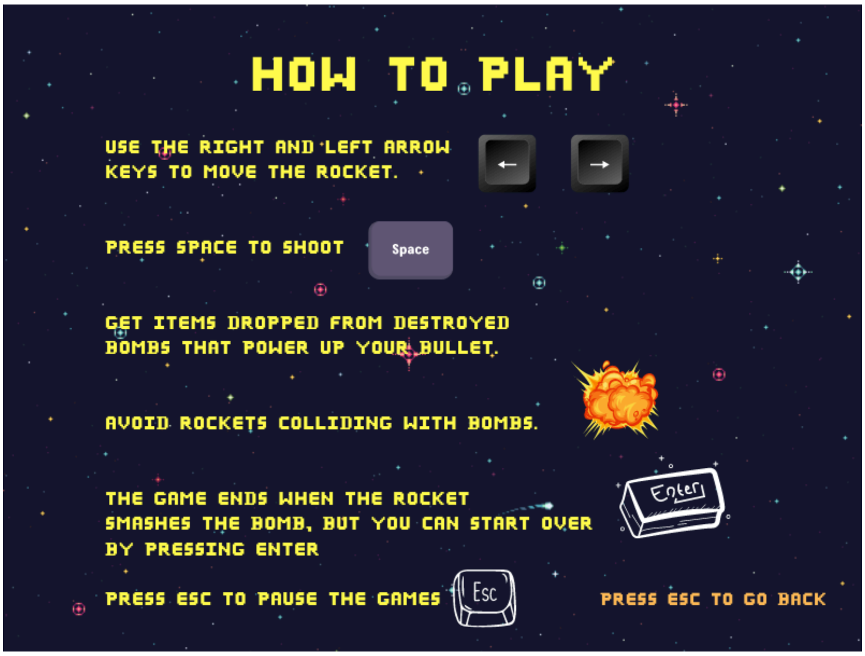
Challenge Yourself

Test your reflexes and strategic skills in "Space Dev" Master the art of dodging enemy attacks while aiming precisely to obliterate incoming threats. How long can you survive in this intense interstellar battle?

Prepare for an adrenaline-pumping journey through the cosmos in "Space Dev" where every moment counts, and every shot could be the difference between victory and defeat. Get ready to embark on an unforgettable space odyssey filled with action, challenge, and endless fun!







HOW TO PLAY

USE THE RIGHT AND LEFT ARROW KEYS TO MOVE THE ROCKET.

PRESS SPACE TO SHOOT

GET ITEMS DROPPED FROM DESTROYED BOMBS THAT POWER UP YOUR BULLET.

AVOID ROCKETS COLLIDING WITH BOMBS.

THE GAME ENDS WHEN THE ROCKET SMASHES THE BOMB, BUT YOU CAN START OVER BY PRESSING ENTER

PRESS ESC TO PAUSE THE GAMES

PRESS ESC TO GO BACK

Key Features

- Player Rocket: Take control of a customizable rocket with different power-up abilities like speed boosts, spread shots, and more.
- Enemy Bombs: Face off against various types of enemy bombs, each with unique behavior and health levels.
- Power-Ups: Collect items dropped by defeated enemies to gain temporary upgrades and enhance your firepower.
- Explosive Action: Witness intense explosion animations and effects as you engage in thrilling combat.
- Progressive Difficulty: As you play, the game's difficulty increases, providing an ever-challenging experience.

Objective

Your goal is to survive for as long as possible by avoiding enemy bombs and shooting them down. With each bomb destroyed, you earn points, and collecting power-ups enhances your capabilities. Can you achieve the highest score and become the ultimate space pilot?

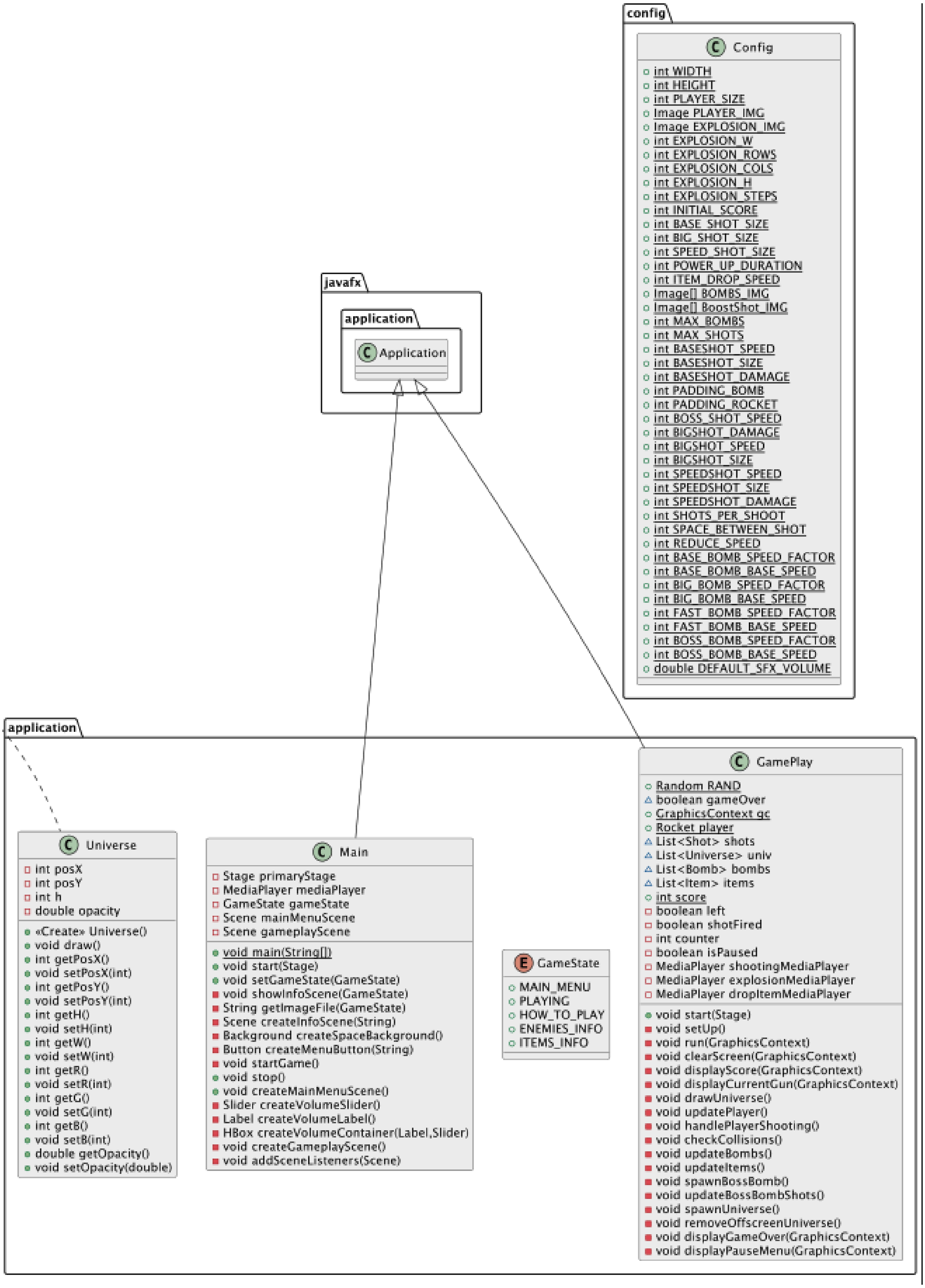
Controls

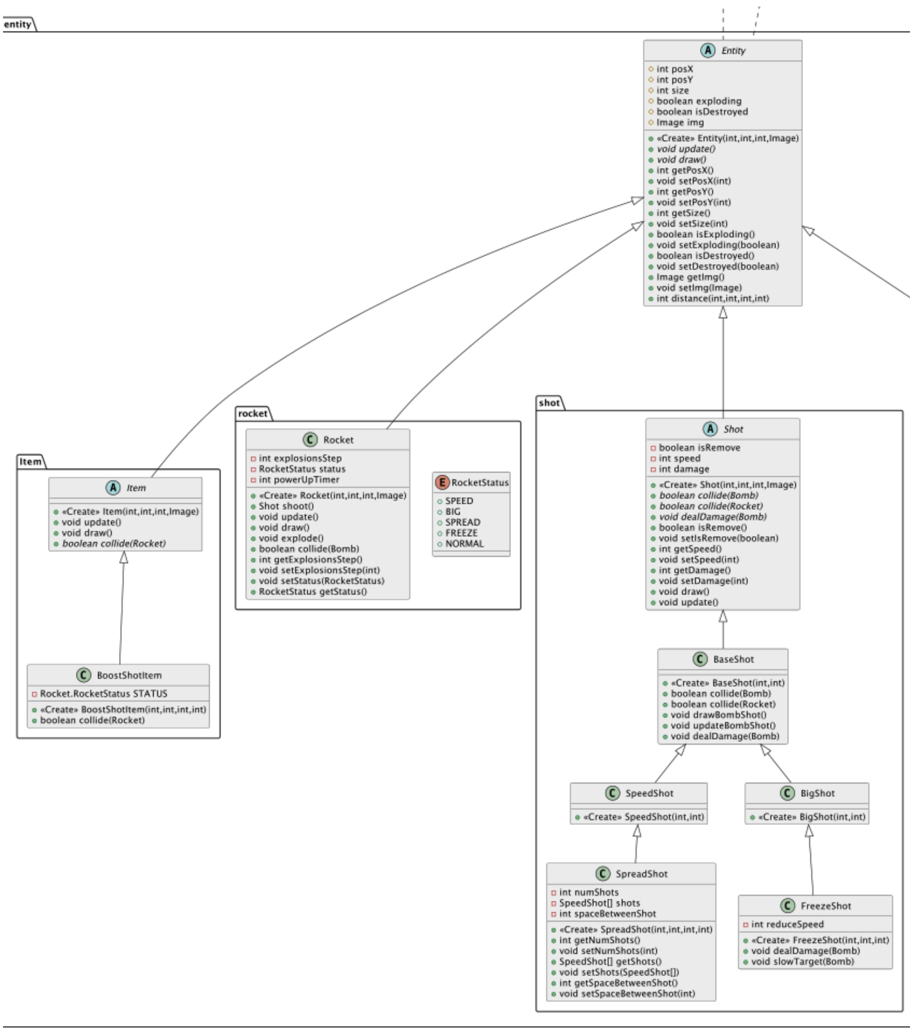
- Use the **Right and Left arrow** keys to **move** the rocket.
- Press **SPACE** to **shoot** the bullet at the bomb.
- Get **items** dropped from destroyed bombs that **power up** your bullet.
- Avoid rockets **colliding** with **bombs**.
- The **game ends** when the rocket smashes the bomb, **but** you can start over by pressing **Enter**
- Press **ESC** to **pause** the games

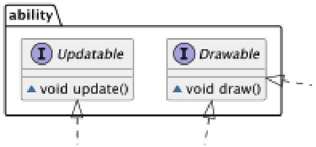
Graphics and Sound

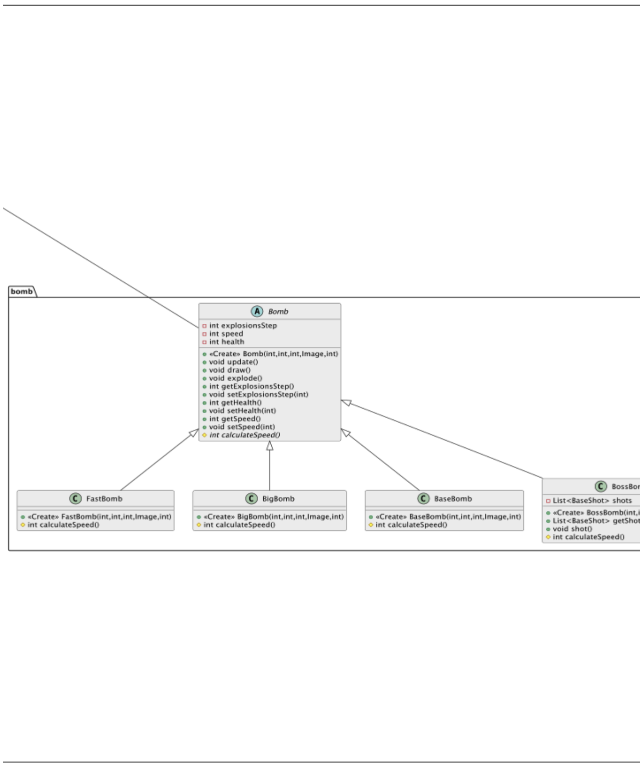
Enjoy immersive visuals with vibrant space-themed graphics, dynamic universe elements, and explosive effects. Experience an engaging atmosphere with accompanying sound effects and music that heighten the excitement of the gameplay.

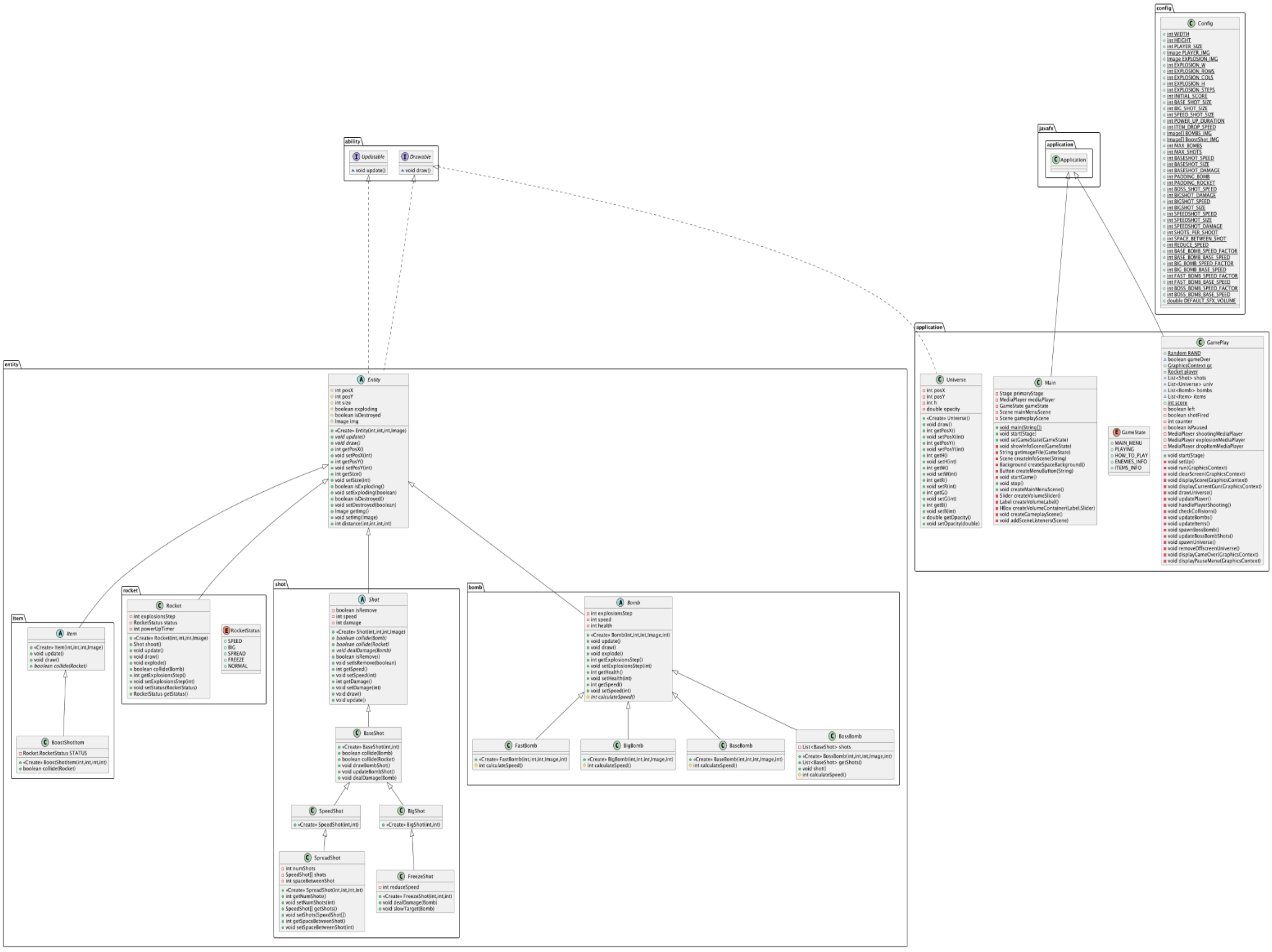
UML


config
Config
int WIDTH
int HEIGHT
int PLAYER_SIZE
Image PLAYER_IMG
Image EXPLOSION_IMG
int EXPLOSION_W
int EXPLOSION_ROWS
int EXPLOSION_COLS
int EXPLOSION_H
int EXPLOSION_STEPS
int INITIAL_SCORE
int BASE_SHOT_SIZE
int BIG_SHOT_SIZE
int SPEED_SHOT_SIZE
int POWER_UP_DURATION
int ITEM_DROP_SPEED
Image[] BOMBS_IMG
Image[] BoostShot_IMG
int MAX_BOMBS
int MAX_SHOTS
int BASESHOT_SPEED
int BASESHOT_SIZE
int BASESHOT_DAMAGE
int PADDING_BOMB
int PADDING_ROCKET
int BOSS_SHOT_SPEED
int BIGSHOT_DAMAGE
int BIGSHOT_SPEED
int BIGSHOT_SIZE
int SPEEDSHOT_SPEED
int SPEEDSHOT_SIZE
int SPEEDSHOT_DAMAGE
int SHOTS_PER_SHOOT
int SPACE_BETWEEN_SHOT
int REDUCE_SPEED
int BASE_BOMB_SPEED_FACTOR
int BASE_BOMB_BASE_SPEED
int BIG_BOMB_SPEED_FACTOR
int BIG_BOMB_BASE_SPEED
int FAST_BOMB_SPEED_FACTOR
int FAST_BOMB_BASE_SPEED
int BOSS_BOMB_SPEED_FACTOR
int BOSS_BOMB_BASE_SPEED
double DEFAULT_SFX_VOLUME
javafx
application
Application
application
Universe
int posX
int posY
int h
double opacity
«Create» Universe()
void draw()
int getPosX()
void setPosX(int)
int getPosY()
void setPosY(int)
int getH()
void setH(int)
int getW()
void setW(int)
int getR()
void setR(int)
int getG()
void setG(int)
int getB()
void setB(int)
double getOpacity()
void setOpacity(double)
Main
Stage primaryStage
MediaPlayer mediaPlayer
GameState gameState
Scene mainMenuScene
Scene gameplayScene
void main(String[])
void start(Stage)
void setGameState(GameState)
void showInfoScene(GameState)
String getImageFile(GameState)
Scene createInfoScene(String)
Background createSpaceBackground()
Button createMenuButton(String)
void startGame()
void stop()
void createMainMenuScene()
Slider createVolumeSlider()
Label createVolumeLabel()
HBox createVolumeContainer(Label,Slider)
void createGameplayScene()
void addSceneListeners(Scene)
GameState
MAIN_MENU
PLAYING
HOW_TO_PLAY
ENEMIES_INFO
ITEMS_INFO
GamePlay
Random RAND
boolean gameOver
GraphicsContext gc
Rocket player
List<Shot> shots
List<Universe> univ
List<Bomb> bombs
List<Item> items
int score
boolean left
boolean shotFired
int counter
boolean isPaused
MediaPlayer shootingMediaPlayer
MediaPlayer explosionMediaPlayer
MediaPlayer dropItemMediaPlayer
void start(Stage)
void setUp()
void run(GraphicsContext)
void clearScreen(GraphicsContext)
void displayScore(GraphicsContext)
void displayCurrentGun(GraphicsContext)
void drawUniverse()
void updatePlayer()
void handlePlayerShooting()
void checkCollisions()
void updateBombs()
void updateItems()
void spawnBossBomb()
void updateBossBombShots()
void spawnUniverse()
void removeOffscreenUniverse()
void displayGameOver(GraphicsContext)
void displayPauseMenu(GraphicsContext)


entity
Entity
int posX
int posY
int size
boolean exploding
boolean isDestroyed
Image img
«Create» Entity(int,int,int,Image)
void update()
void draw()
int getPosX()
void setPosX(int)
int getPosY()
void setPosY(int)
int getSize()
void setSize(int)
boolean isExploding()
void setExploding(boolean)
boolean isDestroyed()
void setDestroyed(boolean)
Image getImg()
void setImg(Image)
int distance(int,int,int,int)
item
Item
«Create» Item(int,int,int,Image)
void update()
void draw()
boolean collide(Rocket)
BoostShotItem
Rocket.RocketStatus STATUS
«Create» BoostShotItem(int,int,int,int)
boolean collide(Rocket)
rocket
Rocket
int explosionsStep
RocketStatus status
int powerUpTimer
«Create» Rocket(int,int,int,Image)
Shot shoot()
void update()
void draw()
void explode()
boolean collide(Bomb)
int getExplosionsStep()
void setExplosionsStep(int)
void setStatus(RocketStatus)
RocketStatus getStatus()
RocketStatus
SPEED
BIG
SPREAD
FREEZE
NORMAL
shot
Shot
boolean isRemove
int speed
int damage
«Create» Shot(int,int,int,Image)
boolean collide(Bomb)
boolean collide(Rocket)
void dealDamage(Bomb)
boolean isRemove()
void setIsRemove(boolean)
int getSpeed()
void setSpeed(int)
int getDamage()
void setDamage(int)
void draw()
void update()
BaseShot
«Create» BaseShot(int,int)
boolean collide(Bomb)
boolean collide(Rocket)
void drawBombShot()
void updateBombShot()
void dealDamage(Bomb)
SpeedShot
«Create» SpeedShot(int,int)
BigShot
«Create» BigShot(int,int)
SpreadShot
int numShots
SpeedShot[] shots
int spaceBetweenShot
«Create» SpreadShot(int,int,int,int)
int getNumShots()
void setNumShots(int)
SpeedShot[] getShots()
void setShots(SpeedShot[])
int getSpaceBetweenShot()
void setSpaceBetweenShot(int)
FreezeShot
int reduceSpeed
«Create» FreezeShot(int,int,int)
void dealDamage(Bomb)
void slowTarget(Bomb)


ability
Updatable
void update()
Drawable
void draw()


bomb
Bomb
int explosionsStep
int speed
int health
«Create» Bomb(int,int,int,Image,int)
void update()
void draw()
void explode()
int getExplosionsStep()
void setExplosionsStep(int)
int getHealth()
void setHealth(int)
int getSpeed()
void setSpeed(int)
int calculateSpeed()
FastBomb
«Create» FastBomb(int,int,int,Image,int)
int calculateSpeed()
BigBomb
«Create» BigBomb(int,int,int,Image,int)
int calculateSpeed()
BaseBomb
«Create» BaseBomb(int,int,int,Image,int)
int calculateSpeed()
BossBor
List<BaseShot> shots
«Create» BossBomb(int,i
List<BaseShot> getShot
void shot()
int calculateSpeed()


config
Config
int WIDTH
int HEIGHT
int PLAYER_SIZE
Image PLAYER_IMG
Image EXPLOSION_IMG
int EXPLOSION_W
int EXPLOSION_ROWS
int EXPLOSION_COLS
int EXPLOSION_H
int EXPLOSION_STEPS
int INITIAL_SCORE
int BASE_SHOT_SIZE
int BIG_SHOT_SIZE
int SPEED_SHOT_SIZE
int POWER_UP_DURATION
int ITEM_DROP_SPEED
Image[] BOMBS_IMG
Image[] BoostShot_IMG
int MAX_BOMBS
int MAX_SHOTS
int BASESHOT_SPEED
int BASESHOT_SIZE
int BASESHOT_DAMAGE
int PADDING_BOMB
int PADDING_ROCKET
int BOSS_SHOT_SPEED
int BIGSHOT_DAMAGE
int BIGSHOT_SPEED
int BIGSHOT_SIZE
int SPEEDSHOT_SPEED
int SPEEDSHOT_SIZE
int SPEEDSHOT_DAMAGE
int SHOTS_PER_SHOOT
int SPACE_BETWEEN_SHOT
int REDUCE_SPEED
int BASE_BOMB_SPEED_FACTOR
int BASE_BOMB_BASE_SPEED
int BIG_BOMB_SPEED_FACTOR
int BIG_BOMB_BASE_SPEED
int FAST_BOMB_SPEED_FACTOR
int FAST_BOMB_BASE_SPEED
int BOSS_BOMB_SPEED_FACTOR
int BOSS_BOMB_BASE_SPEED
double DEFAULT_SFX_VOLUME
javafx
application
Application
ability
Updatable
void update()
Drawable
void draw()
application
Universe
int posX
int posY
int h
double opacity
«Create» Universe()
void draw()
int getPosX()
void setPosX(int)
int getPosY()
void setPosY(int)
int getH()
void setH(int)
int getW()
void setW(int)
int getR()
void setR(int)
int getG()
void setG(int)
int getB()
void setB(int)
double getOpacity()
void setOpacity(double)
Main
Stage primaryStage
MediaPlayer mediaPlayer
GameState gameState
Scene mainMenuScene
Scene gameplayScene
void main(String[])
void start(Stage)
void setGameState(GameState)
void showInfoScene(GameState)
String getImageFile(GameState)
Scene createInfoScene(String)
Background createSpaceBackground()
Button createMenuButton(String)
void startGame()
void stop()
void createMainMenuScene()
Slider createVolumeSlider()
Label createVolumeLabel()
HBox createVolumeContainer(Label,Slider)
void createGameplayScene()
void addSceneListeners(Scene)
GameState
MAIN_MENU
PLAYING
HOW_TO_PLAY
ENEMIES_INFO
ITEMS_INFO
GamePlay
Random RAND
boolean gameOver
GraphicsContext gc
Rocket player
List<Shot> shots
List<Universe> univ
List<Bomb> bombs
List<Item> items
int score
boolean left
boolean shotFired
int counter
boolean isPaused
MediaPlayer shootingMediaPlayer
MediaPlayer explosionMediaPlayer
MediaPlayer dropItemMediaPlayer
void start(Stage)
void setUp()
void run(GraphicsContext)
void clearScreen(GraphicsContext)
void displayScore(GraphicsContext)
void displayCurrentGun(GraphicsContext)
void drawUniverse()
void updatePlayer()
void handlePlayerShooting()
void checkCollisions()
void updateBombs()
void updateItems()
void spawnBossBomb()
void updateBossBombShots()
void spawnUniverse()
void removeOffscreenUniverse()
void displayGameOver(GraphicsContext)
void displayPauseMenu(GraphicsContext)
entity
Entity
int posX
int posY
int size
boolean exploding
boolean isDestroyed
Image img
«Create» Entity(int,int,int,Image)
void update()
void draw()
int getPosX()
void setPosX(int)
int getPosY()
void setPosY(int)
int getSize()
void setSize(int)
boolean isExploding()
void setExploding(boolean)
boolean isDestroyed()
void setDestroyed(boolean)
Image getImg()
void setImg(Image)
int distance(int,int,int,int)
item
Item
«Create» Item(int,int,int,Image)
void update()
void draw()
boolean collide(Rocket)
BoostShotItem
Rocket.RocketStatus STATUS
«Create» BoostShotItem(int,int,int,int)
boolean collide(Rocket)
rocket
Rocket
int explosionsStep
RocketStatus status
int powerUpTimer
«Create» Rocket(int,int,int,Image)
Shot shoot()
void update()
void draw()
void explode()
boolean collide(Bomb)
int getExplosionsStep()
void setExplosionsStep(int)
void setStatus(RocketStatus)
RocketStatus getStatus()
RocketStatus
SPEED
BIG
SPREAD
FREEZE
NORMAL
shot
Shot
boolean isRemove
int speed
int damage
«Create» Shot(int,int,int,Image)
boolean collide(Bomb)
boolean collide(Rocket)
void dealDamage(Bomb)
boolean isRemove()
void setIsRemove(boolean)
int getSpeed()
void setSpeed(int)
int getDamage()
void setDamage(int)
void draw()
void update()
BaseShot
«Create» BaseShot(int,int)
boolean collide(Bomb)
boolean collide(Rocket)
void drawBombShot()
void updateBombShot()
void dealDamage(Bomb)
SpeedShot
«Create» SpeedShot(int,int)
BigShot
«Create» BigShot(int,int)
SpreadShot
int numShots
SpeedShot[] shots
int spaceBetweenShot
«Create» SpreadShot(int,int,int,int)
int getNumShots()
void setNumShots(int)
SpeedShot[] getShots()
void setShots(SpeedShot[])
int getSpaceBetweenShot()
void setSpaceBetweenShot(int)
FreezeShot
int reduceSpeed
«Create» FreezeShot(int,int,int)
void dealDamage(Bomb)
void slowTarget(Bomb)
bomb
Bomb
int explosionsStep
int speed
int health
«Create» Bomb(int,int,int,Image,int)
void update()
void draw()
void explode()
int getExplosionsStep()
void setExplosionsStep(int)
int getHealth()
void setHealth(int)
int getSpeed()
void setSpeed(int)
int calculateSpeed()
FastBomb
«Create» FastBomb(int,int,int,Image,int)
int calculateSpeed()
BigBomb
«Create» BigBomb(int,int,int,Image,int)
int calculateSpeed()
BaseBomb
«Create» BaseBomb(int,int,int,Image,int)
int calculateSpeed()
BossBomb
List<BaseShot> shots
«Create» BossBomb(int,int,int,Image,int)
List<BaseShot> getShots()
void shot()
int calculateSpeed()

2. Implementation Details

To complete this assignment, you need to understand Object-Oriented Programming principles. We will provide details of the class implementation.

* Note that Access Modifier Notations can be listed below
+ (public)
# (protected)
- (private)
underlined (static)
ALL_CAPS (final variable)
italic (abstract)

2.1 Package entity

2.1.1 Abstract Class Entity

The Entity class is an abstract base class for all entities. It implements the Drawable and Updatable interfaces.

Fields

Name	Description
# int posX	An integer representing the X-coordinate of the entity's position.
# int posY	An integer representing the Y-coordinate of the entity's position.
# int size	An integer representing the size of the entity.
# boolean exploding	A boolean indicating whether the entity is currently exploding.
# boolean isDestroyed	A boolean indicating whether the entity is destroyed.
# Image img	object representing the visual representation of the entity.

Constructors

Name	Description
+ Entity(int posX, int posY, int size, Image image)	Create a new instance.

Method

Name	Description
+ *void update()*	abstract method to update the state of the entity.
+ *void draw()*	abstract method to draw the entity on the screen.
+ int getPosX()	Returns X-coordinate position.
+ int getPosX()	Returns Y-coordinate position.

+ void setPosX(int posX)	Sets X-coordinate of the entity.
+ void setPosY(int posY)	Sets Y-coordinate of the entity.
+ int getSize()	Returns the size of the entity.
+ void setSize(int size)	Sets the size of the entity.
+ boolean isExploding()	Returns current state is exploding or not.
+ void setExploding(boolean exploding)	Sets whether the entity is currently exploding.
+ boolean isDestroyed()	Returns current state is destroyed or not.
+ void setIsDestroyed(boolean isDestroyed)	Sets whether the entity is currently destroyed.
+ Image getImg()	Returns image of the entity.
+ int distance(int x1, int y1, int x2, int y2)	Calculates the distance by using pythagoras theorem.

2.1.2 Package entity.rocket

2.1.2.1 Class Rocket

The Rocket class represents the player-controlled rocket in the game. It extends the Entity class.

Fields

Name	Description
- int explosionStep	An integer representing the current step of the explosion animation.
+ enum RocketStatus	An enum RocketStatus representing the current status of the rocket. SPEED, BIG, SPREAD, FREEZE, NORMAL.
- int powerUpTime	An integer representing the duration of power-up effects.

Constructors

Name	Description
+ Rocket(int posX, int posY, int size, Image image)	Create a new instance of Rocket.

Method

Name	Description
+ Shot shoot()	Creates and returns a shot based on the current status of the rocket.
+ void update()	Updates the state of the rocket, including handling power-up duration and explosion animation.
+ void draw()	Draws the rocket on the game canvas, considering its

	current state (normal or exploding).
+ void explode()	Initiates the explosion of the rocket.
+ boolean collide(Bomb)	Checks collision between the rocket and a bomb.
+ int getExplosionsStep()	Retrieves the current step of the explosion animation.
+ void setExplosionsStep(int explosionStep)	Sets the step of the explosion animation.
+ void setStatus(status RocketStatus)	Sets the status of the rocket along with resetting the power-up timer.
+ RocketStatus getStatus()	Retrieves the current status of the rocket.

2.1.3 Package entity.bomb

2.1.3.1 Abstract Class Bomb

The Bomb class represents the bombs in the game. It extends the Entity class.

Field

Name	Description
- int explosionStep	An integer representing the current step of the explosion animation.
- int speed	An integer representing the speed of the bomb.
- int health	An integer representing the health of the bomb.

Constructors

Name	Description
+ Bomb(int posX, int posY, int size, Image image, int health)	Create a new instance of Bomb.

Method

Name	Description
+ void update()	Abstract method to update the state of the entity.
+ void draw()	Abstract method to draw the entity on the screen.
+ void explode()	Makes the bomb start exploding.
+ int getExplosionsStep()	Returns current explosion step.
+ void setExplosionsStep(int explosionsStep)	Sets current explosion step.

+ int getHealth()	Returns health of bomb.
+ void setHealth(int health)	Sets health of bomb.
+ int getSpeed()	Returns speed of bomb.
+ void setSpeed(int speed)	Sets speed of bomb.
# *int calculateSpeed()*	Calculates the speed of the base bomb based on the score and speed factor.

2.1.3.2 Class BaseBomb

The BaseBomb class is one of the bombs in the game.

Constructors

Name	Description
+ BaseBomb(int posX, int posY, int size, Image image, int health)	Create a new instance of BaseBomb.

Method

Name	Description
# int calculateSpeed()	Calculates the speed of the base bomb based on the score and speed factor.

2.1.3.3 Class FastBomb

The FastBomb class is one of the bombs in the game.

Constructors

Name	Description
+ FastBomb(int posX, int posY, int size, Image image, int health)	Create a new instance of FastBomb.

Method

Name	Description
# int calculateSpeed()	Calculates the speed of the fast bomb based on the score and speed factor.

2.1.3.4 Class BigBomb

The BigBomb class is one of the bombs in the game.

Constructors

Name	Description
+ BigBomb(int posX, int posY, int size, Image image, int health)	Create a new instance of BigBomb.

Method

Name	Description
# int calculateSpeed()	Calculates the speed of the big bomb based on the score and speed factor.

2.1.3.5 Class BossBomb

The BossBomb class is one of the bombs in the game.

Field

Name	Description
- List<BaseShot> shots	A list to store the shots fired by the boss bomb.

Constructors

Name	Description
+ BossBomb(int posX, int posY, int size, Image image, int health)	Create a new instance of BossBomb.

Method

Name	Description
+ List<BaseShot> getShots()	Returns the list of shots fired by the boss bomb.
+ void shot()	Creates and adds a new BaseShot to the shots list.
# int calculateSpeed()	Calculates the speed of the boss bomb based on the score and speed factor.

2.1.4 Package entity.shot

2.1.4.1 Abstract Class Shot

The Shot class serves as a base class for different types of shots in the game. It contains common attributes and methods that are shared among all types of shots. It extends the Entity class.

Field

Name	Description
- boolean isRemove	A boolean indicating whether the shot should be removed from the game.
- int speed	An integer representing the speed of the shot.
- int damage	An integer representing the damage inflicted by the shot on entities it collides with.

Constructors

Name	Description
+ Shot(int posX, int posY, int size, Image image, int health)	Create a new instance of Shot.

Method

Name	Description
+ *boolean collide(Bomb bomb)*	An abstract method for checking collision between the shot and a bomb.
+ *boolean collide(Rocket rocket)*	An abstract method for checking collision between the shot and a rocket.
+ *void dealDamage(Bomb bomb)*	An abstract method for dealing damage to a bomb upon collision.
+ boolean isRemove()	A getter method to retrieve the status of whether the shot should be removed.
+ void setIsRemove(boolean isRemove)	A setter method to set the status of whether the shot should be removed.
+ int getSpeed()	A getter method to retrieve the speed of the shot.
+ void setSpeed(int speed)	A setter method to set the speed of the shot.
+ int getDamage()	A getter method to retrieve the damage inflicted by the shot.
+ void setDamage(int damage)	A setter method to set the damage inflicted by the shot.
+ void draw()	An overridden method to draw the shot on the game canvas.
+ void update()	An overridden method to update the state of the shot, such as its position, based on its speed.

2.1.4.2 Class BaseShot

The BaseShot class represents a basic type of shot fired by the player in the game. It extends the Shot class and provides specific implementations for its methods.

Constructors

Name	Description
+ BaseShot(int posX, int posY)	Create a new instance of BaseShot.

Method

Name	Description
+ boolean collide(Bomb bomb)	Overrides the method from the superclass to check collision between the shot and a bomb.
+ boolean collide(Rocket rocket)	Overrides the method from the superclass to check collision between the shot and a rocket.
+ void drawBombShot()	Draw the shot with a yellow oval on the game canvas. This method is specific to the behavior of this type of shot.
+ void updateBombShot()	Updates the position of the shot, moving it upwards with a speed determined by the predefined constants.
+ void dealDamage(Bomb bomb)	Deals damage to the specified bomb when the shot collides with it.

2.1.4.3 Class BigShot

The BigShot class represents a type of shot fired by the player in the game. It is characterized by its larger size and potentially increased damage compared to a regular shot. This class extends the BaseShot class.

Constructors

Name	Description
+ BigShot(int posX, int posY)	Create a new instance of BigShot.

2.1.4.4 Class FreezeShot

The FreezeShot class represents a special type of shot fired by the player in the game. It has the ability to freeze or slow down its target upon collision, reducing its movement speed. This class extends the BigShot class and introduces additional functionality to slow down targets upon dealing damage.

Field

Name	Description
- int reduceSpeed	An integer representing the amount by which the target's speed is reduced upon being hit by the FreezeShot.

Constructors

Name	Description
+ FreezeShot(int posX, int	Create a new instance of FreezeShot.

posY, int reduceSpeed)	

Method

Name	Description
+ void dealDamage(Bomb bomb)	Overrides the method from the superclass to deal damage to the specified bomb upon collision with the FreezeShot. Additionally, it slows down the target by invoking the slowTarget() method.
+ void slowTarget(Bomb bomb)	Slows down the specified bomb's movement speed by the amount specified in the reduceSpeed field. If the resulting speed is less than or equal to 0, it sets the speed to 1 to prevent it from becoming negative. This method ensures that the bomb's updated speed is applied.

2.1.4.5 Class SpeedShot

The SpeedShot class represents a type of shot fired by the player in the game with enhanced speed. It inherits from the BaseShot class and provides specific implementations for its methods to accommodate the behavior of a fast-moving shot.

Constructors

Name	Description
+ SpeedShot(int posX, int posY)	Create a new instance of SpeedShot.

2.1.4.6 Class SpreadShot

The SpreadShot class represents a type of shot fired by the player in the game, which releases multiple shots simultaneously in a spread pattern. It inherits from the SpeedShot class and provides additional functionality to manage the spread of shots.

Field

Name	Description
- int numShots	An integer representing the number of shots fired in the spread pattern.
- SpeedShot[] shots	An array of SpeedShot objects representing the individual shots in the spread.
- int spaceBetweenShot	An integer representing the space between each shot in the spread pattern.

Constructors

Name	Description

+ SpreadShot(int posX, int posY, int numShots, int spaceBetweenShot)	Create a new instance of SpreadShot.

Method

Name	Description
+ int getNumShots()	Retrieves the number of shots in the spread pattern.
+ void setNumShots(int numShots)	Sets the number of shots in the spread pattern.
+ SpeedShot[] getShots()	Retrieves the array of individual shots in the spread pattern.
+ void setShots(SpeedShot[] shots)	Sets the array of individual shots in the spread pattern.
+ int getSpaceBetweenShot()	Retrieves the space between each shot in the spread pattern.
+ void setSpaceBetweenShot(int spaceBetweenShot)	Sets the space between each shot in the spread pattern.

2.1.5 Package entity.item

2.1.5.1 Abstract Class Item

The Item class represents a generic item in the game that can be collected or interacted with by the player. It serves as a base class for specific types of items that provide various effects or bonuses to the player when collected. This class contains common attributes and methods shared among all types of items.

Constructors

Name	Description
+ Item(int posX, int posY, int size, Image image)	Create a new instance of Item.

Method

Name	Description
+ void update()	Overrides the method from the superclass to update the state of the item. By default, it handles the vertical movement of the item, simulating its descent or appearance on the game screen. It also checks if the item has moved below the game screen boundary and marks it as destroyed if so.
+ void draw()	Overrides the method from the superclass to draw the item on the game canvas. If the item is not destroyed, it is drawn at its current position

	using its image.
+ boolean collide(Rocket rocket)	An abstract method for checking collision between the item and a rocket. Subclasses must implement this method to define specific collision behavior between the item and the player's rocket.

2.1.5.2 Class BoostShotItem

The BoostShotItem class represents an item in the game that boosts the player's rocket with a specific status when collected. It inherits from the Item class and provides functionality to apply a boost to the rocket's status upon collision with the player's rocket.

Field

Name	Description
- Rocket.RocketStatus STATUS	An enumeration representing the rocket status that will be applied when the boost shot item is collected.

Constructors

Name	Description
+ BoostShotItem(int posX, int posY, int size, int type)	Create a new instance of BoostShotItem. The type parameter determines the rocket status that will be applied when the boost shot item is collected.

Method

Name	Description
+ boolean collide(Rocket rocket)	Overrides the method from the superclass to handle collision between the boost shot item and the player's rocket. If a collision occurs, the boost shot item is destroyed, and the rocket's status is updated to the specified status associated with the item. Returns true if a collision occurs, otherwise false.

2.2 Package config

2.2.1 Class Config

The config package encapsulates various constants and configurations used throughout the game. Here's a description of its contents:

Field

Name	Description
+ int WIDTH	The width of the game window.

+ int HEIGHT	The height of the game window.
+ int PLAYER_SIZE	The size of the player entity.
+ Image PLAYER_IMG	The image used for the player entity.
+ Image EXPLOSION_IMG	The image used for the explosion animation.
+ int EXPLOSION_W	The width of each explosion frame.
+ int EXPLOSION_ROWS	The number of rows in the explosion sprite sheet.
+ int EXPLOSION_COLS	The number of columns in the explosion sprite sheet.
+ int EXPLOSION_H	The height of each explosion frame.
+ int EXPLOSION_STEPS	The total number of steps in the explosion animation.
+ int INITIAL_SCORE	The initial score of the player.
+ BASE_SHOT_SIZE	The size of the base shot.
+ int BIG_SHOT_SIZE	The size of the big shot.
+ int SPEED_SHOT_SIZE	The size of the speed shot.
+ int POWER_UP_DURATION	The duration of the power-up effect in frames.
+ int ITEM_DROP_SPEED	The speed at which items drop.
+ Image[] BOMBS_IMG	An array of images used for different bomb types.
+ Image[] BoostShot_IMG	An array of images used for boost shot items.
+ int MAX_BOMBS	The maximum number of bombs that can be present on the screen.
+ int MAX_SHOTS	The maximum number of shots that can be present on the screen.
+ int BASESHOT_SPEED	The speed of the base shot.
+ int BASESHOT_SIZE	The size of the base shot.
+ int BASESHOT_DAMAGE	The damage dealt by the base shot.
+ int PADDING_BOMB	The padding between bombs.
+ int PADDING_ROCKET	The padding between rocket and screen.
+ int BOSS_SHOT_SPEED	The speed of the boss's shot.
+ int BIGSHOT_DAMAGE	The damage dealt by the big shot.
+ int BIGSHOT_SPEED	The speed of the big shot.
+ int BIGSHOT_SIZE	The size of the big shot.
+ int SPEEDSHOT_SPEED	The speed of the speed shot.
+ int SPEEDSHOT_SIZE	The size of the speed shot.

+ int SPEEDSHOT_DAMAGE	The damage dealt by the speed shot.
+ int SHOTS_PER_SHOOT	The number of shots per shooting action.
+ int SPACE_BETWEEN_SHOT	The space between each shot.
+ int REDUCE_SPEED	The amount by which the speed is reduced.
+ int BASE_BOMB_SPEED_FACTOR	The speed factor for the base bomb.
+ int BASE_BOMB_BASE_SPEED	The base speed for the base bomb.
+ int BIG_BOMB_SPEED_FACTOR	The speed factor for the big bomb.
+ int BIG_BOMB_BASE_SPEED	The base speed for the big bomb.
+ int FAST_BOMB_SPEED_FACTOR	The speed factor for the fast bomb.
+ int FAST_BOMB_BASE_SPEED	The base speed for the fast bomb.
+ int BOSS_BOMB_SPEED_FACTOR	The speed factor for the boss bomb.
+ int BOSS_BOMB_BASE_SPEED	The base speed for the boss bomb.
+ DEFAULT_SFX_VOLUME	The base sfx sound volume

2.3 Package application

2.3.1 Class Universe

The Universe class represents celestial objects or phenomena in the game environment. These objects are typically background elements that add visual interest and ambiance to the game. The class implements the Drawable interface, indicating that instances of Universe can be drawn on the game canvas.

Field

Name	Description
- int posX	The x-coordinate position of the universe object.
- int posY	The y-coordinate position of the universe object.
- int h	The height of the universe object.
- int w	The width of the universe object.
- int r	The red component of the color of the universe object.
- int g	The green component of the color of the universe object.
- int b	The blue component of the color of the universe object.

- double opacity	The opacity of the universe object, determining its transparency.

Constructors

Name	Description
+ Universe()	Initializes a new instance of the Universe class with random values for its position, size, color, and opacity. This constructor is responsible for generating varied and dynamic universe objects.

Method

Name	Description
+ void draw()	Draws the universe object on the game canvas. It sets the color and opacity of the object based on its attributes and fills an oval shape at its position. Additionally, it updates the position of the object to create a scrolling effect, simulating movement in the game environment.
+ int getPosX()	Retrieves the x-coordinate position of the universe object.
+ void setPosX(int posX)	Sets the x-coordinate position of the universe object to the specified value.
+ int getPosY()	Retrieves the y-coordinate position of the universe object.
+ void setPosY(int posY)	Sets the y-coordinate position of the universe object to the specified value.
+ int getH()	Retrieves the height of the universe object.
+ void setH(int h)	Sets the height of the universe object to the specified value.
+ int getW()	Retrieves the width of the universe object.
+ void setW(int w)	Sets the width of the universe object to the specified value.
+ int getR()	Retrieves the red component of the color of the universe object.
+ void setR(int r)	Sets the red component of the color of the universe object to the specified value.
+ int getG()	Retrieves the green component of the color of the universe object.
+ void setG(int g)	Sets the green component of the color of the universe object to the specified value.
+ int getB()	Retrieves the blue component of the color of the universe object.
+ void setB(int b)	Sets the blue component of the color of the universe object to the specified value.
+ int getOpacity()	Retrieves the opacity of the universe object.

+ void setOpacity(int opacity)	Sets the opacity of the universe object to the specified value.

2.3.2 Class GamePlay

The **GamePlay** class extends the **Application** class from JavaFX and represents the main gameplay logic of the game. It handles the game loop, user input, and manages the game entities such as the player, shots, bombs, and items.

Fields

Name	Description
- Random RAND	A static final Random object used for generating random values.
- boolean gameOver	Indicates whether the game is over or not.
- GraphicsContext gc	A static GraphicsContext object used for drawing on the canvas.
- Rocket player	A static Rocket object representing the player's rocket.
- List<Shot> shots	A list of Shot objects representing the shots fired by the player.
- List<Universe> univ	A list of Universe objects representing the game universe.
- List<Bomb> bombs	A list of Bomb objects representing the bombs in the game.
- List<Item> items	A list of Item objects representing the items in the game.
- int score	A static integer representing the player's score.
- boolean left	Indicates whether the left key is pressed.
- boolean right	Indicates whether the right key is pressed.
- boolean shoot	Indicates whether the shoot key is pressed.
- boolean restart	Indicates whether the restart key is pressed.
- boolean shotFired	Indicates whether a shot has been fired.
- int counter	A counter used for tracking game events.
- boolean isPaused	Indicates whether the game is paused or not.
- MediaPlayer shootingMediaPlayer	A MediaPlayer object for playing the shooting sound effect.
- MediaPlayer explosionMediaPlayer	A MediaPlayer object for playing the explosion sound effect.

- MediaPlayer dropItemMediaPlayer	A MediaPlayer instance for playing the item drop sound effect.

Methods

Name	Description
+ void start(Stage stage)	Overrides the start method from the Application class. It sets up the game window, canvas, and event handlers for user input. It also initializes the game entities and starts the game loop.
- void setUp()	Sets up the initial state of the game by initializing the game entities, score, and sound effects.
- void run(GraphicsContext gc)	The main game loop method that is called repeatedly. It updates the game state, renders the game entities, and handles collisions and game over conditions.
- void clearScreen(GraphicsContext gc)	Clears the screen by filling it with a gray color.
- void displayScore(GraphicsContext gc)	Displays the player's score on the screen.
- void displayCurrentGun(GraphicsContext gc)	Displays the player's current gun status on the screen.
- void drawUniverse()	Draws the background elements (Universe objects) on the screen.
- void updatePlayer()	Updates the player's position based on user input and draws the player on the screen.
- void handlePlayerShooting()	Handles the player's shooting action based on user input and adds shots to the shots list.
- void checkCollisions()	Checks for collisions between game entities such as shots, bombs, and the player. It updates the game state accordingly, including scoring and item drops.
- void updateBombs()	Updates the bombs' positions, handles their destruction, and spawns new bombs.
- void updateItems()	Updates the positions of collectible items, handles their collection by the player, and plays sound effects.

- void spawnBossBomb()	Spawns a boss bomb when certain conditions are met.
- void updateBossBombShots()	Updates the shots fired by the boss bomb and checks for collisions with the player.
- void spawnUniverse()	Spawns new background elements (Universe objects) based on a random chance.
- void removeOffscreenUniverse()	Removes background elements (Universe objects) that have moved off the screen.
- void displayGameOver(GraphicsContext gc)	Displays the game over screen with the player's score and instructions to restart the game.
- void displayPauseMenu(GraphicsContext gc)	Displays the pause menu when the game is paused.

2.3.3 Class Main

The **Main** class is the entry point of the game application. It **extends** the **Application** class from JavaFX and is responsible for setting up the main menu and starting the game.

Fields

Name	Description
- Stage primaryStage	The primary stage of the game application.
- MediaPlayer mediaPlayer	The media player for playing background music.
- GameState gameState	The current state of the game.
- Scene mainMenuScene	The scene for the main menu.
- Scene gameplayScene	The scene for the gameplay.
+ enum GameState	The state for currently, it can be MAIN_MENU, PLAYING, HOW_TO_PLAY, ENEMIES_INFO, ITEMS_INFO

Methods

Name	Description
+ void main(String[] args)	The main method that launches the game application.
+ void start(Stage primaryStage)	Overrides the start method from the Application class. Sets up the primary stage, creates the main menu and gameplay scenes, and sets the initial game state to **MAIN MENU**

- void setGameState(GameState state)	Sets the current game state and updates the scene accordingly. If the state is **MAIN_MENU**, it sets the scene to the main menu scene and creates a new gameplay scene. If the state is **PLAYING**, it sets the scene to the gameplay scene and starts the game. For other states, it shows the corresponding info scene.
- void showInfoScene(GameState state)	Shows the info scene based on the provided game state.
- String getImageFile(GameState state)	Returns the image file path based on the provided game state.
- Scene createInfoScene(String imageFile)	Creates and returns an info scene with the provided image file. It creates a StackPane layout, loads the image, creates an ImageView, and sets up the content layout with the image view.
- Background createSpaceBackground()	Creates and returns a background image for the main menu.
- Button createMenuButton(String text)	Creates and returns a menu button with the provided text. It sets the button's dimensions, font, text color, and styles for default and hover states.
- void startGame()	Starts the gameplay by creating an instance of the `GamePlay` class and setting up the gameplay scene.
+ void stop()	Overrides the stop method from the Application class. Stops the background music when the game is closed.
+ void createMainMenuScene()	Creates the main menu scene with background music, menu buttons, and layout.
- Slider createVolumeSlider()	Creates and returns a volume slider. It sets the slider's range, tick unit, tick count, and adds a listener to update the media player's volume when the slider value changes.
- Label createVolumeLabel()	Creates and returns a volume label with white text color.
- HBox createVolumeContainer(Label volumeLabel, Slider volumeSlider)	Creates and returns an HBox container for the volume label and slider. It sets the spacing and alignment of the container.
+ void createGameplayScene()	Creates the gameplay scene with a canvas and layout.
- void addSceneListeners(Scene scene)	Adds key event listeners to the provided scene. If the Escape key is pressed, it sets the game state to **MAIN_MENU**.

Link Youtube Video : https://www.youtube.com/watch?v=74KAzyjKe34