Earthlingz Documentation

Created by

Pranesh Ingkanunt 6531322921

Pathorn Roungtan 6531333821

2110215 Programming Methodology

Semester 2 Year 2022

Chulalongkorn University

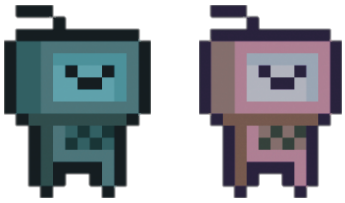
Earthlingz

Introduction

EarthlingZ is inspired by “Worms Armageddon” by Team17 and other Platformer games. Each team will take turns trying to defeat all the other teams and be the last team remaining. Characters or “Earthlings” will shoot various types of rockets to kill their enemies in different ways.

Game Mechanics

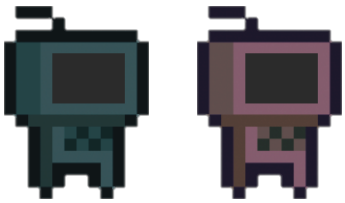
Earthlings



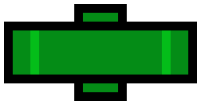
These are “Earthlings”. They can walk, jump, and use their bazooka to shoot rockets freely. But once they have shot a rocket, they need to wait for their next turn. Their enemies are the other color of them.

Earthlings are physics objects. They have physics properties such as mass, velocity, acceleration, decayable velocity, and collider. Their position will be calculated by these properties and collision.

Earthlings start with 100 HP. When their HP drops to 0 or they fall out of the map, they will die and spawn their corpses that will bounce out of the map.



Bazooka



Bazooka is the weapon that earthlings use. It will shoot rockets that earthlings have selected in the direction that point to the mouse cursor. The rocket's velocity depends on the time the player charges before firing.

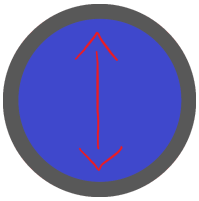
Rockets

Rockets are also physics objects. When rockets collide with other things, they will explode and cause effects depending on their type.
There are 3 types of rocket that Earthlings can use:

1. Normal rocket

When this type of rocket explodes, it will deal high damage to nearby earthlings and push them away with medium force, and destroy all terrains within its explosion radius. Each team has an infinite number of normal rockets.

2. Vertical rocket



This type of rocket will explode in a vertical line, destroying any terrains above and below while dealing very low damage to nearby earthlings, and pushing them away with little force. Each team has only 2 vertical rockets.

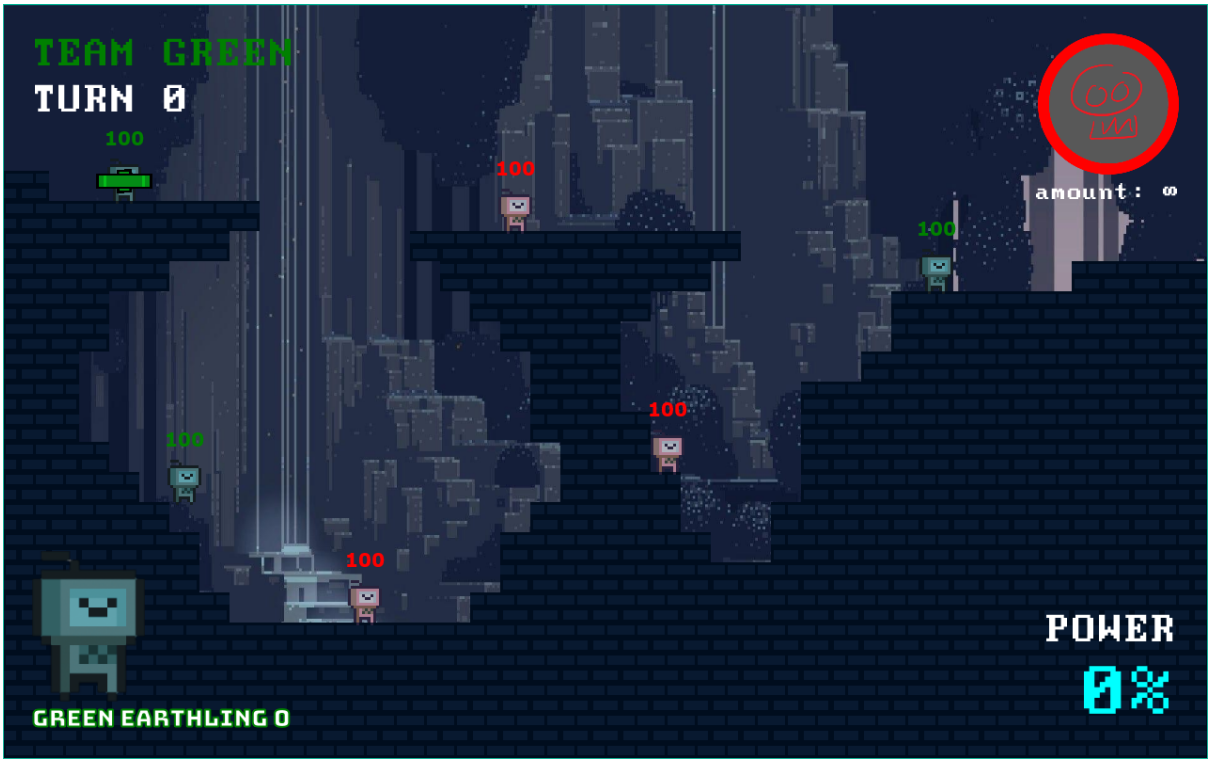
3. Push rocket

When this type of rocket explodes, it will deal low damage to nearby earthlings and push them away with strong force. However, this type of rocket will not destroy terrains. Each team has 10 push rockets.

Floor box

Floor box is a physics object that will not move. Earthlings can stand on it and it can be destroyed by rockets (except push rockets).

Game Map



This is the game map, when the game starts, there will be an amount of earthling and floor box spawned depending on the map that is played. However, the game currently has only one map, “Ancient”.

Controls Summary

PRESS A	**MOVE LEFT**
PRESS D	**MOVE RIGHT**
PRESS SPACEBAR	**JUMP**
PRESS RIGHT CLICK	**CHANGE SELECTED ROCKET TYPE**
PRESS AND HOLD LEFT CLICK	**CHARGE THE ROCKET**
RELEASE LEFT CLICK	**SHOOT THE ROCKET**

Gameplay

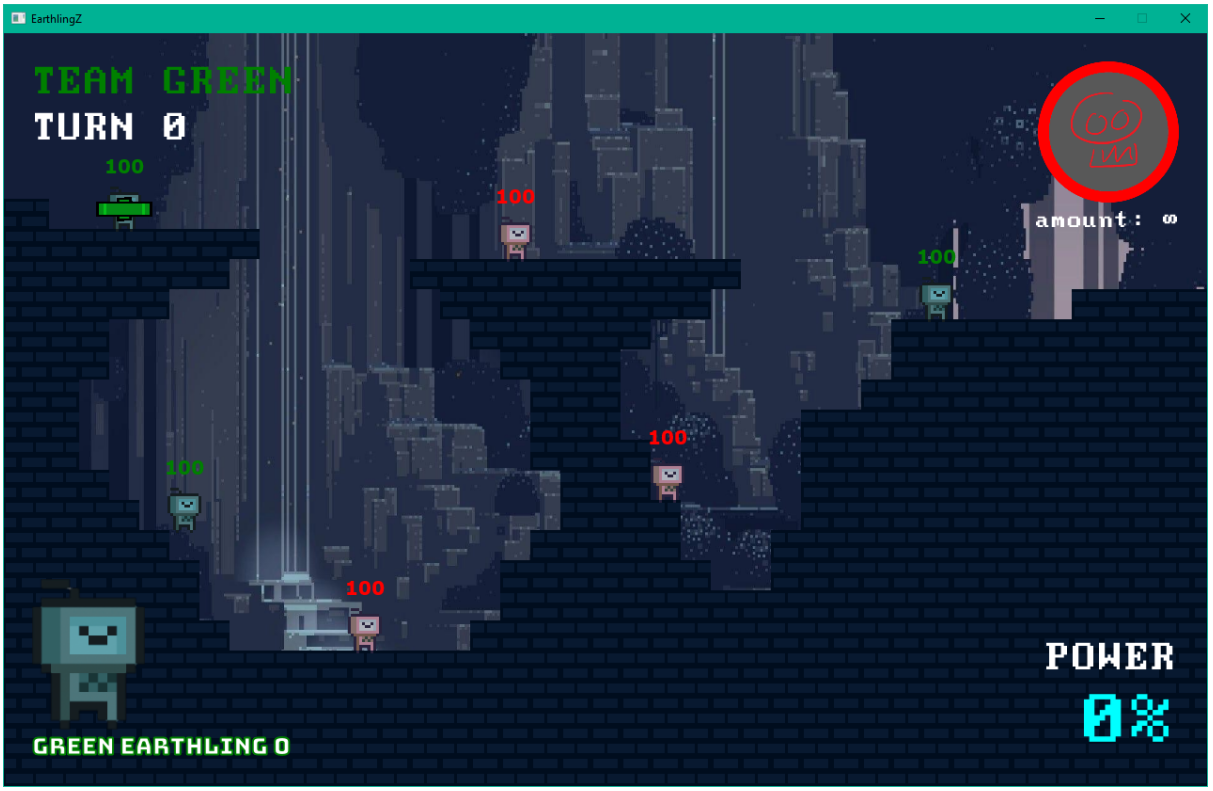
Title Scene

The game will start with the title scene. This scene contains the game title, start button, and exit button.

When the player presses the start button, the game will change to the gameplay scene and gameplay will start.

When the player presses the exit button, the game will stop.

Gameplay Scene



This is where the gameplay starts.

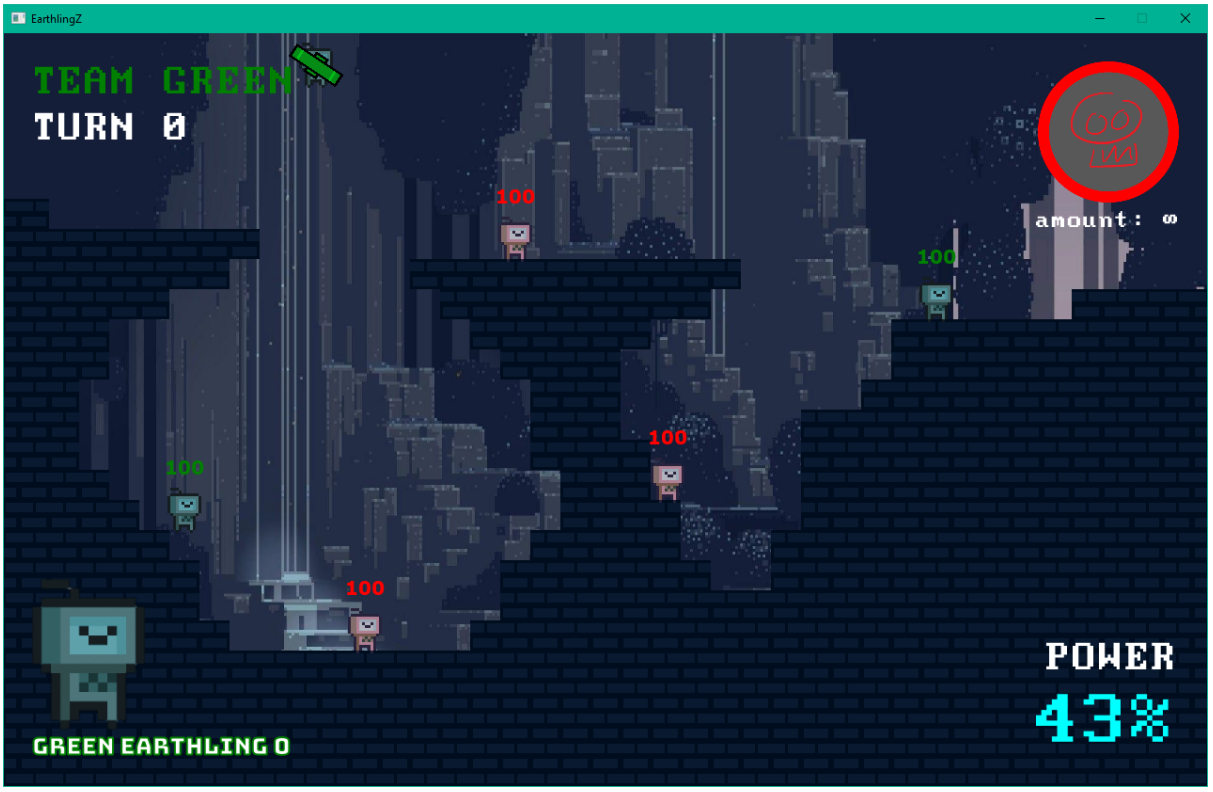
Background, floor boxes, and all earthlings will be spawned according to the Game Map. (currently is “AncientMap”)

Current team and turn number will be shown at the top left corner.

Information of the earthling that the player is controlling appears at the bottom left corner.

Selected rocket and amount left of that type of rocket will be shown at the top right corner.

Current rocket charge percentage will be shown at the bottom right corner.



The game starts with Team Green's turn.

The earthling that the player is controlling will equip a bazooka which will point toward the mouse cursor.

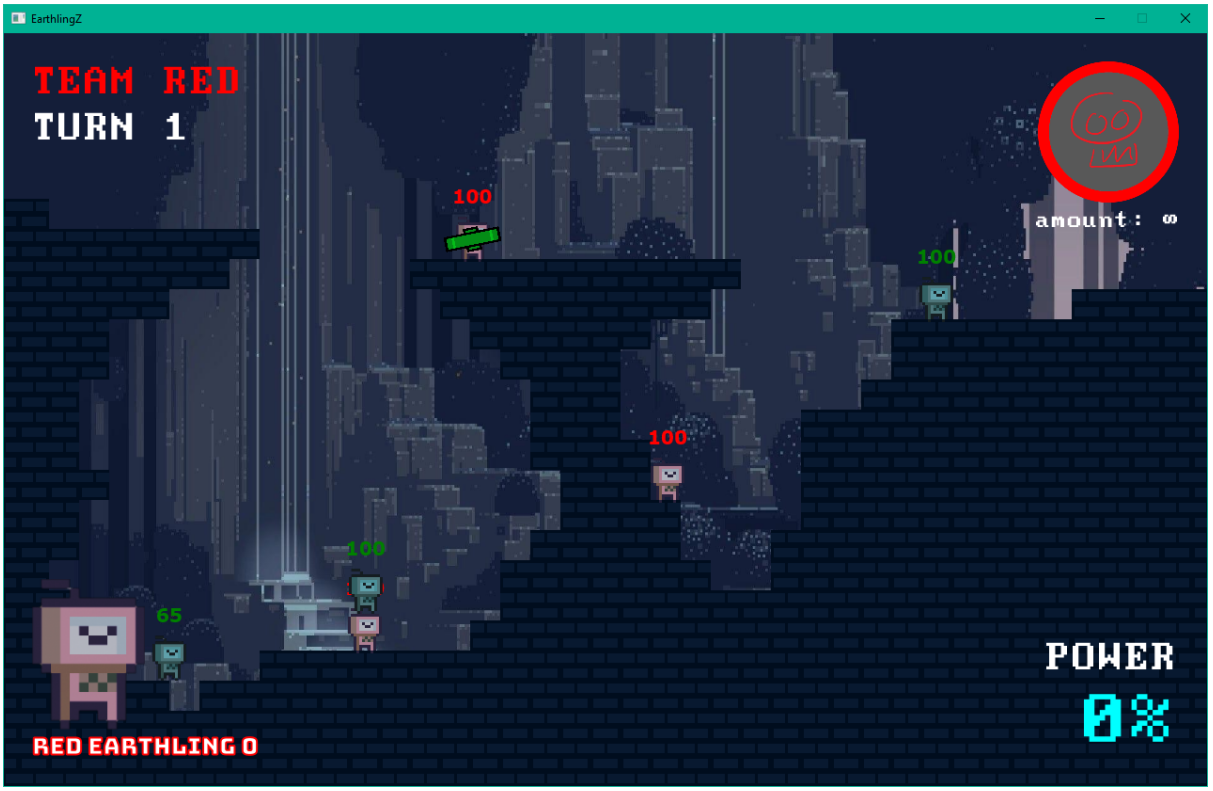
Players can move the earthling horizontally by pressing ‘A’(move left) and ‘D’(move right).

When the player presses ‘SPACEBAR’ when the earthling is on the ground, the earthling will jump.

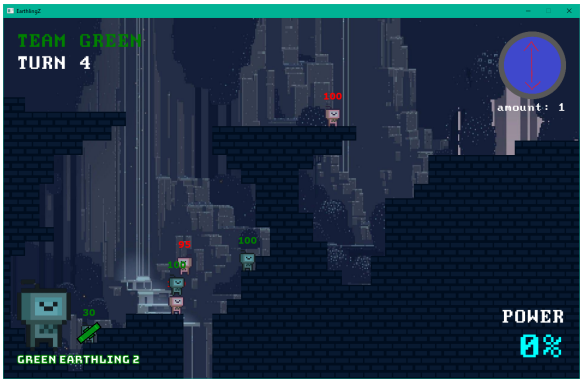
The player can press ‘Right Click’ to change selected rocket type in the cycle of Normal Rocket, Vertical Rocket, and Push Rocket. (the rocket that has already run out will be skipped.)

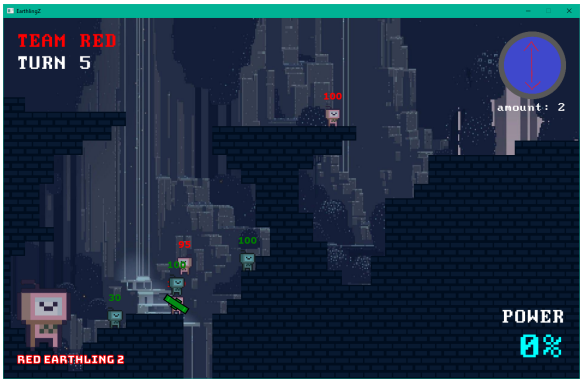
The player needs to hold ‘Left Click’ to charge the rocket and release ‘Left Click’ to shoot it.

The rocket’s velocity depends on the time the player charges before firing and is shown in the current rocket charge percentage (POWER 43%).



After the rocket has been shot, the team will switch so as the turn. Next player will control their team's earthlings. The next earthling that each player gets to control will be a cycle for each of its team. But if the next earthling is dead from the last turn, their turn will be skipped.





The amount of each type of rocket left will be separated for each team. (In above picture, Team Green has used 1 Vertical Rocket but Team Red doesn't)

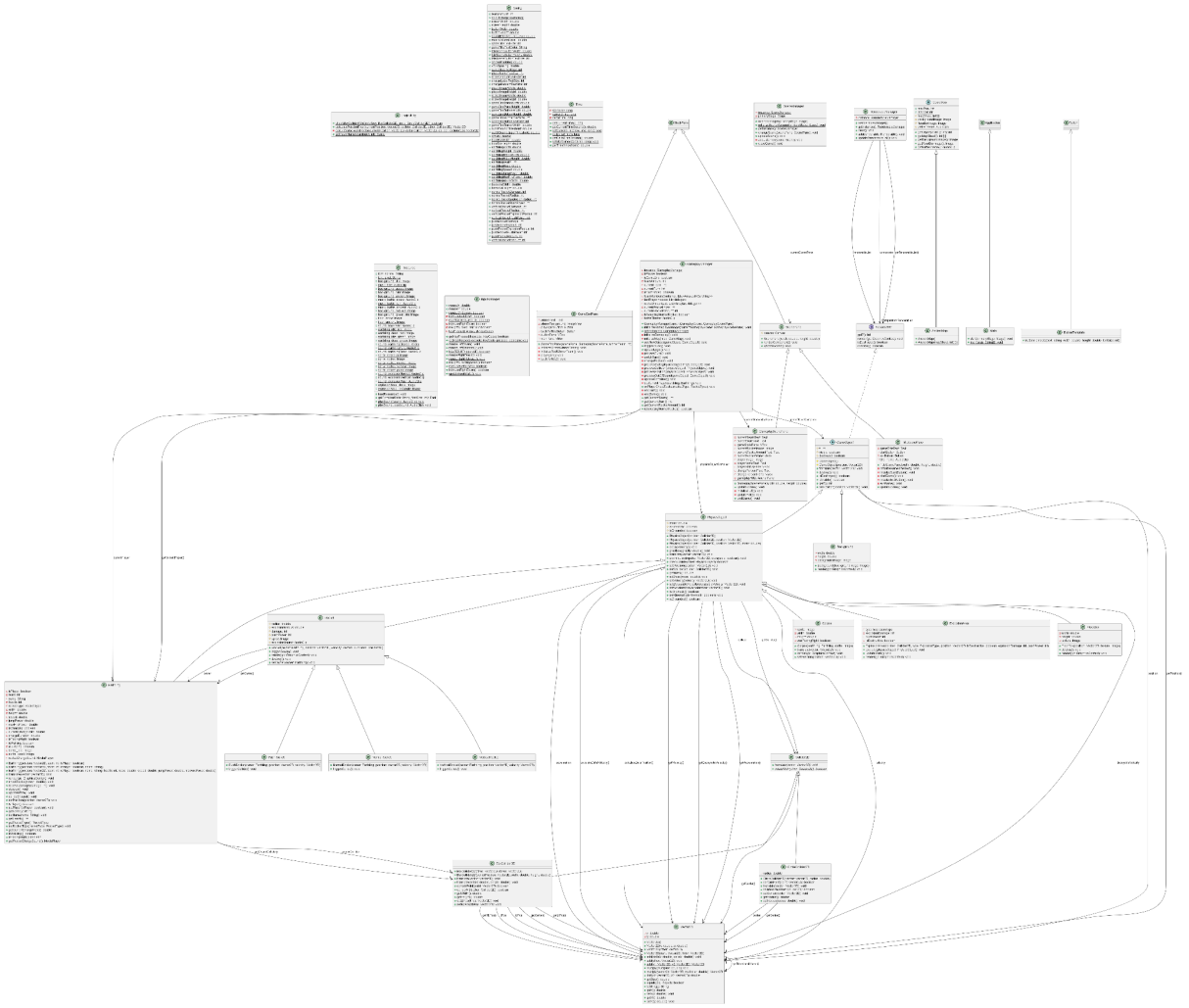
When there is only one team left, that team will win and the Game Ending Pane will show up.

The Game Ending Pane contains the winner team name and their background, the play again button, and the back to title button.

When the player presses the play again button, the gameplay will reset.

When the player presses the back to title button, the game will change to the title scene.

Class diagram



Source codes

Access Modifier Notations

+	public
#	protected
-	private
underlined	static
italic	abstract

1. Package application

Contains main class to run the application

1. public class Main

   Initializes stage, SceneManager, Time, and other game logics in AnimationTimer

2. Package config

Contains configuration settings

1. public class Config

   Contains configures variables (public static final) that is used in the application

3. Package gameObject

Contains classes for game objects

1. public abstract class GameObject implements Renderable

   An object that can be rendered.

   Field

# Vector2D position	Represents the object's position
# int z	Represents the order that it will be rendered
# boolean visible	Shows if the object is visible (visible object will be rendered)
# boolean destroyed	Shows if the object is destroyed(destroyed object will be removed)

   Constructor

# GameObject()	Constructs new GameObject with position (0,0)
# GameObject(Vector2D position)	Constructs new GameObject with given position

Method

+ void translate(Vector2D vector)	Changes the object's position according to given vector
+ void destroy()	Changes the object's destroyed status to true
getters for destroyed, visible, z, position	
setters for position	

2. public abstract class PhysicsObject extends GameObject

A GameObject that has physics properties and can behave in physical ways

Field

# Collider2D collider	Represents the object (PhysicsObject must have Collider2D properties)
# double mass	Represents the object's mass
# Vector2D velocity	Represents the object's velocity
# Vector2D decayableVelocity	Represents the object's decayable velocity
# Vector2D acceleration	Represents the object's acceleration
# boolean isKinematic	States if the object can move
# boolean isGrounded	States if the object is touching

	the ground

Constructor

+ PhysicsObject(Collider2D collider)	Constructs new PhysicsObject with position (0,0)
+ PhysicsObject(Collider2D collide, Vector2D position)	Constructs new PhysicsObject with given position
+ PhysicsObject(Collider2D collide, Vector2D position, double mass)	Constructs new PhysicsObject with given position and mass

Method

+ Vector2D calculateDeltaVelocity()	Calculates the object's changed velocity in the next frame
+ Vector2D calculateDeltaPosition()	Calculates the object's changed position in the next frame
+ void decayVelocity()	Decays the decayable velocity according to its mass
+ void gravitate(double gravity)	Sets an object's acceleration to the gravity constant if it isn't grounded
+ void translate(Vector2D vector)	Changes an object's position according to given vector
+ void addImpulse(Vector2D impulse, boolean decayable)	Changes an object's velocity according to the impulse vector and the object mass
+ boolean checkCollide(PhysicsObject other)	Checks if this object collides with given object
+ void setPosition(Vector2D position)	Translates an object's to a new position
getters and setters for collider, mass, velocity, decayableVelocity,	

acceleration, isKinematic	
getter for isGrounded	

3. public class Background extends GameObject

A background image

Field

- double width	Contains screen's width
- double height	Contains screen's height
- Image backgroundImage	Contains a background image

Constructor

+ Background(Image backgroundImage)	Initializes background image to the screen's width and height set z to -1000

Method

+ render(GraphicsContext gc)	Renders the object

4. public class FloorBox extends PhysicsObject

A floor box (terrain of the map)

Field

- double width	Contains floor box's width
- double height	Contains floor box's height
- Image texture	Contains a floor box texture

Constructor

+ FloorBox(Vector2D position, Image texture)	Initializes floor box texture to BoxCollider2D with its width and height set z to 5, set isKinematic to false

Method

+ void destroy()	Destroys the floor box and plays the sound effect when it breaks
+ void render(GraphicsContext gc)	Renders the object

5. public class Earthling extends PhysicsObject

Earthling character

Field

- enum RocketType	Contains 3 types of rockets
- boolean isPlayer	Checks if it's player's turn
- int team	earthling's team
- int health	earthling's health
- rocketType	earthling's selected rocket type
- double width	earthling's width
- double height	earthling's height
- Boxcollider2D groundCollider	Acts as earthling's hitbox
- double speed	earthling's speed
- double jumpPower	earthling's jump power
- double maxFirePower	earthling's max rocket charge power
- boolean isCharging	Checks if player is charging

	bazooka
- double currentChargeRate	Current bazooka charge rate
- double chargeDuration	Bazooka charging duration
- boolean isFacingRight	Checks if the character is facing right
- boolean isWalking	Checks if the character is walking
- boolean isJumping	Checks if the character is jumping
- Image sprite_idle	Character's normal/idle sprite image
- Image sprite_dead	Character's dead sprite image

Constructor

+ Earthling(Vector2D position, int team, boolean isPlayer)	Constructs earthling according to configuration
+ Earthling(Vector2D position, int team, boolean isPlayer, String name)	Constructs earthling with given name
+ Earthling(Vector2D position, int team, boolean isPlayer, String name, int health, double mass, double speed, double jumpPower, double maxFirePower)	Constructs earthling with given name and additional parameters

Method

+ void translate(Vector2D vector)	Changes object's position according to given vector
+ render(GraphicsContext gc)	Renders the earthling according to direction it is facing then renders the bazooka if the

	earthling is player after that draw earthling's hp above it
+ void shootRocket(double power)	Shoots a rocket according to given power
+ void receiveDamage(int damage)	Minus earthling's health by damage and plays its hurt sound effect. If health is 0 or below, destroys it
+ void destroy()	Destroys the object and plays its dead sound effect, then create its corpse
+ void updateState()	Updates object's state based on keyboard and mouse input
+ void collideGround()	If the object hits the ground, sets its status to grounded and not jumping, and its velocity and acceleration on Y-axis to 0 if its more than 0
+ void setPosition(Vector2D position)	Sets the object's position to the new position
getters and setters for isPlayer, name, rocketType	
getters for health, groundCollider, currentChargeRate, isWalking, isFacingRight, rocketChargeSound	

6. public class Corpse extends PhysicsObject

Earthling when dead

Field

- Image sprite	Contains corpse's sprite image

- double width	Character's width
- double height	Character's height
- boolean wasFacingRight	Checks if the character was facing right (before dying)

Constructor

+ Corpse(Earthling earthling, Image sprite)	Create new Corpse according to the dead earthling add starting velocity

Method

+ void translate(vector2D vector)	Changes object's position according to the vector
+ void render(GraphicsContext gc)	Renders the object
+ void setPosition(Vector 2D position)	Sets object's position to the new position

4. Package gui

Contains GUI layouts

1. public class ButtonTemplate extends Button

   Sets template of button layout

   Constructor

+ ButtonTemplate(String text, double width, double height, int fontSize)	Sets up button style and layout

2. public class GameEndPane extends StackPane

   Sets up pane after the game is finished

Field

- Text winnerText	Text that shows the winner team
- ImageView winnerBackground	Image that represent the winner team
- Button playAgainButton	Button that press to play again
- Button backToTitleButton	Button that press to back to title
- VBox buttonPane	Vertical box that stores button
- VBox mainPane	Vertical box that stores winnerText and buttonPane

Constructor

+ GameEndPane(Gameplay ScenePane scenePane, int winnerTeam)	Sets up ending scene according to the winner team

Method

- void initializeGreenWinnerTeam()	Sets up the panes according to green team
- void initializeRedWinnerTeam()	Sets up the panes according to red team
- void playAgain()	Creates new gameplay scene
- void backToTitle()	Changes to title scene

5. Package input

Contains input logics for keyboard and mouse

1. public class InputManager

Contains input states

Field

+ double mouseX	mouse cursor position in x

	coordinate
+ double mouseY	mouse cursor position in y coordinate
+ boolean mouseOnScreen	States if the mouse cursor is on screen
- boolean isMouseLeftDown	States if currently left click
- boolean isLeftClickedLastTick	States if left clicked last frame
- boolean isMouseRightDown	States if currently right click
- boolean isRightClickedLastTick	States if right clicked last frame
- ArrayList<KeyCode> keyPressed	List that store all keys that are currently pressed

Method

+ boolean getKeyPressed(KeyCode keycode)	Check that the given key is currently pressed
+ void setKeyPressed(KeyCode keycode, boolean pressed)	Update the state of the given key (pressed or not pressed)
+ void mouseLeftDown()	Set isMouseLeftDown to true Set isLeftClickedLastTick to true
+ void mouseLeftRelease()	Set isMouseLeftDown to false
+ boolean isLeftClickTriggered()	Check that left clicked just last frame
+ void mouseRightDown()	Set isMouseRightDown to true Set isRightClickedLastTick to true
+ void mouseRightRelease()	Set isMouseRightDown to false

+ boolean isRightClickTriggered()	Check that right clicked just last frame
+ boolean isMouseLeftDown()	return isMouseLeftDown
+ boolean isMouseRightDown()	return isMouseRightDown
+ void updateInputState()	Set isLeftClickedLastTick to false Set isRightClickedLastTick to false

6. Package logic

Contains general game logics

1. public abstract class Collider2D

Abstract class for objects with collision

Method

+ *Vector2D getCenter()*	Abstract method Gets object's center position
+ *void translate(Vector2D vector)*	Abstract method Changes its position according to given vector
+ *boolean collideWith(Collider2D other)*	Abstract method Checks if the collider collides with other collider

2. public class BoxCollider2D extends Collider2D

Rectangle collision area

Field

- Vector2D ltPos	Represents top left position of a box
- Vector2D rbPos	Represents bottom right position

	of a box

Constructor

+ BoxCollider2D(Vector2D ltPos, Vector2D rbPos)	Creates a box based on given positions
+ BoxCollider2D(Vector2D centerPosition, double width, double height)	Creates a box by calculating top left position and bottom right position based on given parameters

Method

+ void translate(Vector2D vector)	Moves this object based on given vector
+ void translate(double deltaX, double deltaY)	Moves this object based on X and Y distance
+ boolean containPoint(Vector2D point)	Checks if this object contains given point
+ boolean collideWith(Collider2D other)	Checks if the collider is collided with given collider based on its type
+ Vector2D getCenter()	calculate center position of the box and return
+ double getWidth()	calculate width of the box and return
+ double getHeight()	calculate height of the box and return
getters and setters for ltPos and rbPos	

3. public class CircleCollider2D extends Collider2D

Circle collision area

Field

- Vector2D center	Represents center point of a circle
- double radius	Represents radius of a circle

Constructor

+ CircleCollider2D(Vector2D center, double radius)	Creates a circle based on given center position and radius

Method

+ boolean containPoint(Vector2D point)	Checks if this object contains given point
+ void translate(Vector2D vector)	Moves this object based on given vector
+ boolean collideWith(Collider2D other)	Checks if the collider is collided with given collider based on it type
getters and setters for center and radius	radius cannot be less than 0

4. public class GameplayManager

Contains all game logic and gameplay variables

Field

- <u>GameplayManager instance</u>	current instance of GameplayManager
- GameplayScenePane currentGameplayScene	current gameplay scene pane
- List<GameObject> gameObjectContainer	list of all game objects
- List<PhysicsObject>	list of all physics objects

	physicsObjectContainer	
-	boolean isPause	States that if game is currently pause
-	boolean isGameEnd	States that if gameplay is ended
-	int teamAmount	numbers of team in gameplay
-	int currentTeam	current team index
-	List<ArrayList<Earthling>> teamMembersContainer	list of all earthling for each team
-	List<Integer> lastPlayerIndexes	last player index for each team
-	Earthling currentPlayer	earthling that is current player
-	List<ArrayList<Integer>> rocketAmountList	list of amount for each rocket type for each team
-	int currentRocketIndex	current rocket index
-	int currentRocketAmount	current rocket amount
-	boolean isSelectingNormalRocket;	States that current team is selecting NormalRocket or not
-	AudioClip battleTheme	battle music

Constructor

+	GameplayManager(GameplayScenePane currentGameplayScene)	initialize GameplayManager

Method

+	void initializeGameplayManager(GameplayScenePane currentGameplayScene)	initialize GameplayManager set current gameplay scene
+	GameplayManager getInstance()	get current instance of GameplayManager

- void initializeGameplay()	set up gameplay variable
- void initializeMap(GameMap map)	create object according to given map
+ void addNewObject(GameObject gameObject)	add new object to gameObjectContainer, physicsObjectContainer if it's PhysicsObject, and teamMembersContainer if it's Earthling
+ void togglePause()	toggle Pause or Unpause
+ void updateLogic()	update game logic if game is not pause by - process turn system - check if player is changing rocket - process physics object state, collision, and position(if it's kinematic) - remove game object that is out of screen - update container
- void processTurn()	switch turn if turn end and end game if 1 or less team remaining
- void switchTurn()	switch team to next team assign next earthling to be player
- void changeRocket()	cycle to next rocket(Normal->Vertical->Push->Normal)
- void processState(PhysicsObject physicsObject)	update state of each physics object according to its type
- void processCollision(PhysicsObject physicsObject)	process collision of each physics object according to its type
- void	update position of each kinematic

processPosition(PhysicsObject physicsObject)	physics object according to its type
- void processOutOfBound(GameObject gameObject)	remove object that is out of screen
- void updateContainer()	remove destroyed object
- void setCurrentPlayer(Earthling earthling)	set current player
- void onPlayerShootRocket(RocketType rocketType)	update amount of rocket left in current team and change rocket type if it's 0
- void endTurn()	set isTurnEnded to true
- void endGame()	set isGameEnd to true stop battle music call endGame function of Gameplay Scene Pane stop charging sound
getters for currentTeam, currentTurn, currentPlayer, currentRocketAmount, isSelectingNormalRocket	

7. Package map

Contains map details

1. public abstract class GameMap

Acts as a base for game maps

Field

+ final int mapRow	Sets number of rows to 26
+ final int mapCol	Sets number of columns to 41

# int[ ][ ] mapSheet	Contains arrays of numbers (0 - 2) which represent map layout
# Image backgroundImage	Contains background image for a map
# Image floorBoxImage	Contains floor box image for a map
# AudioClip battleTheme	Contains battle theme music for a map

Method

+ int getObject(int x, int y)	get value in given indexes from mapSheet
getters for mapSheet, backgroundImage, floorBoxImage, and battleTheme	

2. public class AncientMap extends GameMap

Contains information for Ancient Map

Constructor

+ AncientMap()	Create new AncientMap with default mapSheet set backgroundImage, floorBoxImage, and battleTheme according to the map
+ AncientMap(int[ ][ ] mapSheet)	Create new AncientMap with given mapSheet set backgroundImage, floorBoxImage, and battleTheme according to the map

8. Package render

Contains rendering logics

1. public interface Renderable

This interface defines methods and variables for objects that can be rendered.

Method

+ int getZ()	get z value which represent the order that the object will be rendered
+ void render(GraphicsContext gc)	Renders the object
+ boolean isDestroyed()	Checks if the object is destroyed (destroyed object will be removed)
+ boolean isVisible()	Checks if the object is visible (visible object will be rendered)

2. public class RenderableManager

Manage renderable objects

Field

- final RenderableManager instance	instance of RenderManager
- List<Renderable> renderableList	list of renderable objects
- Comparator<Renderable> comparator	operation to compare each pair of members in renderableList

Constructor

+ RenderableManager()	Create new RenderableManager and initialize renderableList and comparator

Method

+ RenderableManager getInstance()	get instance of RenderableManager
+ void clear()	clear renderableList
+ void add(Renderable renderable)	add new renderable object to renderableList and sort
+ void updateRenderableList()	remove destroyed renderable object from renderableList
getter for renderableList	

9. Package rocket

Contains logic for rockets and explosions

1. public abstract class Rocket extends PhysicsObject

Defines general properties of a rocket

Field

# double radius	Radius of the rocket
# double explosionRadius	Radius of the explosion from the rocket
# int damage	Damage of the rocket
# int pushPower	Push power of the rocket, representing how far it can push Earthling
# Earthling owner	Owner of the rocket
# Image sprite	Sprite of the rocket
# AudioClip explosionSound	Explosion sound of the rocket

Constructor

+ Rocket(Earthling owner, Vector2D position, Vector2D velocity, Collider2D collider)	Creates a rocket based on given parameters

Method

+ *void triggerCollide()*	Abstract method When the object is collided, creates an effect
+ render(GraphicsContext gc)	Renders the object
+ void destroy()	Destroys the rocket and plays its sound effect
getter and setter for owner	

2. public class NormalRocket extends Rocket

Properties of a normal rocket

Constructor

+ NormalRocket(Earthling owner, Vector2D position, Vector2D velocity)	Creates a normal rocket based on given parameter

Method

+ void triggerCollide()	Creates a circle explosion area with its parameter , then destroys itself

3. public class PushRocket extends Rocket

Properties of a push rocket

Constructor

+ PushRocket(Earthling	Creates a push rocket based on

owner, Vector2D position, Vector2D velocity)	given parameters and configuration

Method

+ void triggerCollide()	Creates a circle explosion area with its parameter , then destroys itself

4. public class VerticalRocket extends Rocket

Properties of a vertical rocket

Constructor

+ PushRocket(Earthling owner, Vector2D position, Vector2D velocity)	Creates a push rocket based on given parameters and configuration

Method

+ void triggerCollide()	Creates a rectangle explosion area with its parameter , then destroys itself

5. public class ExplosionArea extends PhysicsObject

Creates explosion area for rockets

Field

+ enum ExplosionType	Defines shapes of explosion: circle and rectangle
- ExplosionType type	Contains type of explosion
- int explosionDamage	Value of damage from explosion
- int pushPower	Value of push power from

	explosion
- boolean isDestructive	Defines if the explosion can destroys objects

Constructor

+ ExplosionArea(Collider2D collider, ExplosionType type, Vector2D position, boolean isDestructive, int explosionDamage, int pushPower)	Creates explosion area based on given parameters

Method

+ void explode(PhysicsObject physicsObject)	Creates different effects depend on types of physicsObject, then destroys it
+ void updateState()	Destroys the object
+ void render(GraphicsContext gc)	Renders the object

10. Package scene

Contains logics for different scenes

1. public abstract class ScenePane extends StackPane

Layouts of ScenePane

Field

# Canvas canvas	Provides canvas for a scene

Constructor

+ ScenePane(double width, double height)	Initializes a new scene pane and canvas

Method

+ void renderComponent()	Renders the scene
+ *void updateScene()*	Abstract method Updates the scene

2. public class GameplayScenePane extends ScenePane

Gameplay scene

Field

- Text currentTeamText	Current team text
- Text currentTurnText	Current turn text
- VBox gameStatePane	Pane of game state
- Image currentRocketImage	Image of the current rocket
- Text currentRocketAmountText	Text of the current rocket
- VBox currentRocketPane	Pane of the current rocket
- Image playerImage	Image of the player
- Text playerNameText	Player name
- VBox playerStatusPane	Pane of the player status
- Text chargePercentText	Text of current rocket charge percentage
- VBox chargePercentPane	Pane of the charge percentage
- AnchorPane gameplayHUD	Pane of a gameplay HUD

Constructor

+ GameplayScenePane()	Creates new

double width, double height)	GameplayScenePane, and calls initializeHUD()

Method

+ void updateScene()	Renders scene and updates HUD and update logic from GameplayManager
- void initializeHUD()	Initializes new HUD
- void updateHUD()	Updates HUD
+ void endGame()	Shows game end scene

3. public class TitleScenePane

Field

- Text gameTitleText	Game title
- Button startButton	Start button
- Button exitButton	Exit button
- AudioClip titleTheme	Title Scene music

Constructor

+ TitleScenePane(double width, double height)	Initialize a pane of the title scene

Method

- void initializeGameTitleText()	Initialize title of the game
- void initializeStartButton()	Initialize start button
- void startGame()	Start game
- void initializeExitButton()	Initialize an exit button
- void exitGame()	Exit game

+ void updateScene()	Update scene

4. public class SceneManager

Field

- SceneManager instance	current instance of SceneManager
- final Stage primaryStage	application stage
- ScenePane currentScenePane	Contains current scene pane

Constructor

+ SceneManager(Stage primaryStage)	Setup stage and Shows title scene

Method

+ void initializeSceneManager(Stage primaryStage)	initializes new SceneManager
+ SceneManager getInstance()	returns instance
+ void changeScene(ScenePane scenePane)	changes to another scene
+ void updateScene()	updates the scene
- void addListerner(Scene scene)	adds input listeners to the scene
+ void closeGame()	closes the stage

11. Package utils

Contains logic utility and resources for the application

1. public class LogicUtility

Contains various utility methods

Method

+ boolean checkBoxCollidewithCircle (BoxCollider2D box, CircleCollider2D circle)	Checks if a box collided with a circle
+ Vector2D calculatePositionFixer(Vector2D currentPosition, Collider2D collider, Collider2D other)	Calculates deltaPosition to normalize the earthling position
- Vector2D calculateBackwardVector(Vector2D currentPosition, Vector2D deltaPosition, Collider2D collider)	Calculates backward vector of a collider
+ String getTeamName(int teamIndex)	Returns the team name according to given team index

2. public class Resource

Loads all resource files in public static variable

Method

+ Font getFont(String fontPath, int fontSize)	get font from given path with given size
+ playSound(AudioClip sound)	play sound with the config volume
+ void playSoundLoop(AudioClip sound)	play sound with the config volume and loop

3. public class Time

Contains methods related to time

Field

- long startTime	application start time in nanoseconds
- long currentTime	current application time in nanoseconds
- long deltaTime	time between this frame and last frame in nanoseconds

Method

+ long getCurrentTime()	get current application time in nanoseconds
+ double getCurrentTimeSecond()	get current application time in seconds
+ void setCurrentTime(long newTime)	set current time and calculate delta time
+ long getDeltaTime()	get delta time in nanoseconds
+ double getDeltaTimeSecond()	get delta time in seconds
+ void setStartTime(long startTime)	set startTime
+ double getTimeSinceStart()	get time since application start

4. public class Vector2D

Provides 2-dimensional vector system

Field

- double x	Value in x coordinate of Vector2D
- double y	Value in y coordinate of Vector2D

Constructor

+ Vector2D()	Initializes a new vector to (0, 0)
+ Vector2D(double x, double y)	Initializes a new vector to (x, y)
+ Vector2D(Vector2D other)	Initializes a new vector the same as another vector
+ Vector2D(Vector2D start, Vector2D finish)	Initializes a new vector from starting point to finishing point

Method

+ void add(double deltaX, double deltaY)	Adds this vector by deltaX and deltaY
+ void add(Vector2D other)	Adds this vector with another vector
+ Vector2D add(Vector2D v1, Vector2D v2)	Returns a new vector from adding 2 vectors
+ void multiply(double multiplier)	Multiplies this vector by the multiplier
+ Vector2D multiply(Vector2D vector2d, double multiplier)	Returns a new vector from multiplying a vector to a multiplier
+ double dot(Vector2D v1, Vector2D v2)	Returns dot product of 2 vectors
+ double getsize()	Returns the size of this vector
+ Vector2D getDirectionalVector()	Returns this vector's directional vector

+ boolean equals(Object obj)	Check if this vector is equal to other object
+ String toString()	return in format “(x, y)”
getters and setters for x and y	

Resource files

1. background

   Stores background images

2. entity

   Stores bazooka, rocket, and explosion area sprites

3. font

   Stores fonts

4. music

   Stores background music

5. sound

   Stores sound effects

6. sprite

   Stores earthling sprites

7. texture

   Stores map texture