“Ice War”

Created by

Aunchisa Suwanchatree 6330587521

Tinnakit Udsa 6332018121

Programming Methodology1 2110215

Semester 1 Academic Year 2021

Final Project Documentation

ICE WAR

This game is inspired by Battle City. In the game, you are in the role of a penguin who need to fight with their enemy bear to stay alive. When you have overcome all enemy bears, you advance to the next level with a different layout. In each level, there's some item which you can use to your advantage for passing level easier.

Rules

Character

Main Character - Penguin

The player starts the game with the default penguin, which you can power up to better when collecting the fish and go back to buy the upgrade item in the shop.

Enemy Character – Bear

There are 4 different types of bears. The white bear is the bear that is a short-range attacker like when the player uses the stick. The runner bear is the bear with high speed and short-range attacker. The brown bear is similar to Ice bear but a long-range attacker as the player uses a snowball. The panda bear has very high HP and attack, but will hit slowly.

White Bear

Runner Bear

Brown Bear

Panda Bear

Weapon

There is 2 type of weapon in this game - stick and snow.

The difference is the shooting distance. The stick is a default weapon that can attack short-range damage. The snow fire snowball can take long-range damage

Stick

Snow

Map

The game offers semi-destructible environments that mean there are some obstacles that you can destroy or take advantage of using as a shield from enemies in shorts depends on your strategy. There are three different types of blocks on the map: water, ice, rock.

Picture of blocks are water, ice, and rock block, respectively.

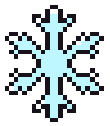
Water is a non-destructible block that the player and enemy cannot be crossed but the snow and stick (bullet) can go through.

Ice is the block can destroy completely, allowing the player to cut a path to its target.

Rock is similar to ice with highly armored.

Items

The items are the random item that can be picked up after destroying the enemy. It helps defeat the enemy by healing the player's health, changing the weapon, freezing the enemy, increasing weapon attack and storing fish for purchasing in the shop to upgrade health or weapon attack.



Freeze Item

The enemy will stop moving a little while

Healing Item

This item is used to heal player's HP.

Snowball Item

This item is used to swap the weapon to snow and after snowball duration expires, Player's weapon will return to default weapon

Fish Item

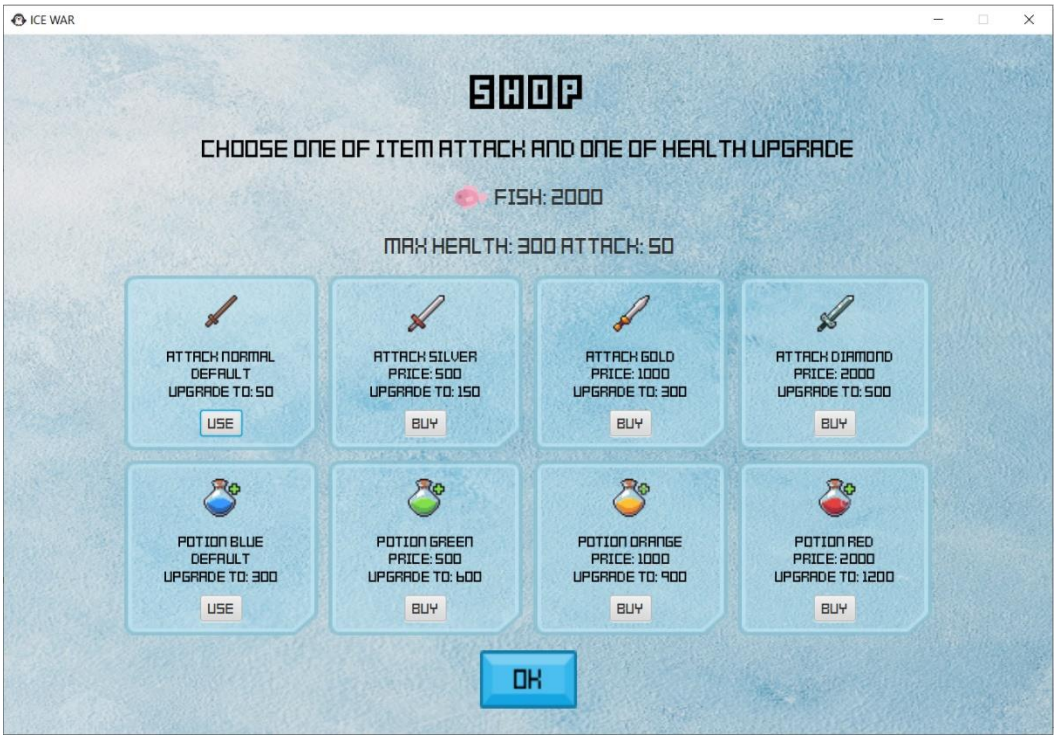
It is working as same as coin in real life. You can collect it to buy upgrading item in shop after finishing that round.

Attack Item

It is used to increase the increases your offensive until end that round.

Shop Item

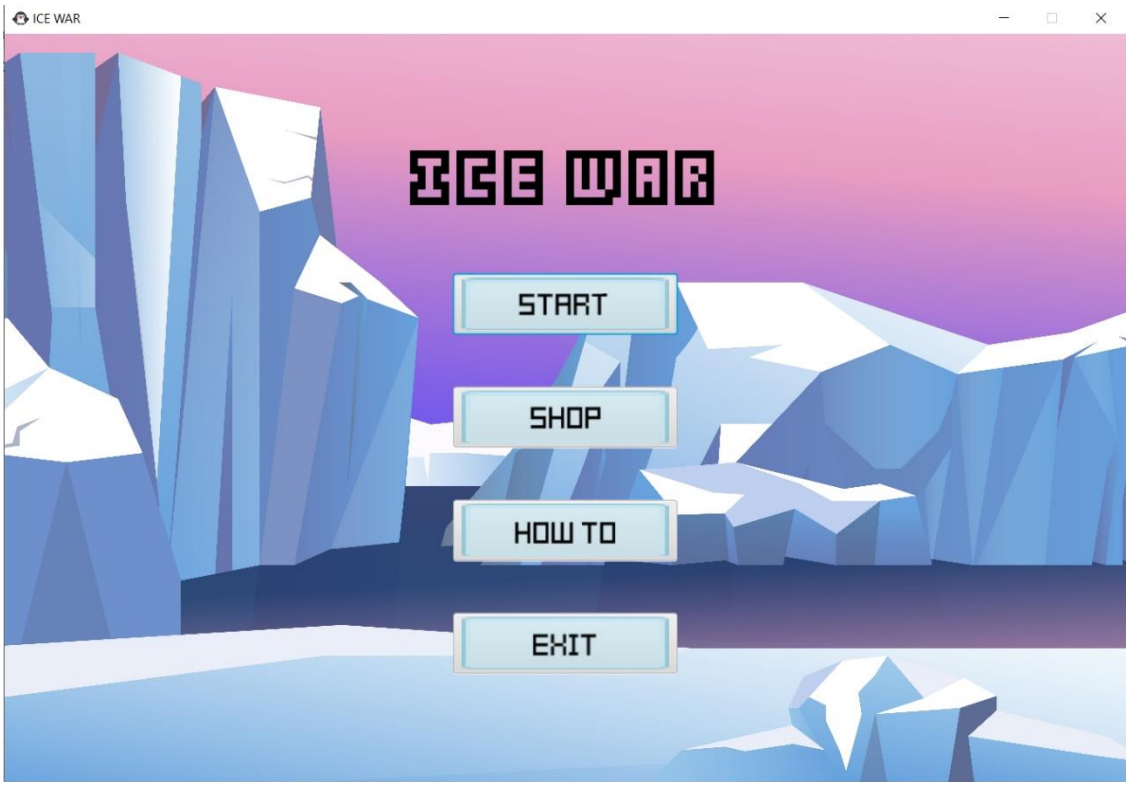
Players can collect the fish in the game to purchase for upgrading power and health. The shop can open in Main Menu which is start before playing new game. There are 4 upgrading level of weapon attack and health potion.



There are 4 different values of upgrading health potion which is Blue Health Potion(default), Green Health Potion, Orange Health Potion and Red Health Potion

There are 4 different values of upgrading weapon attack which is Blue Health Attack Normal, Attack Normal, Attack Normal, Attack Silver

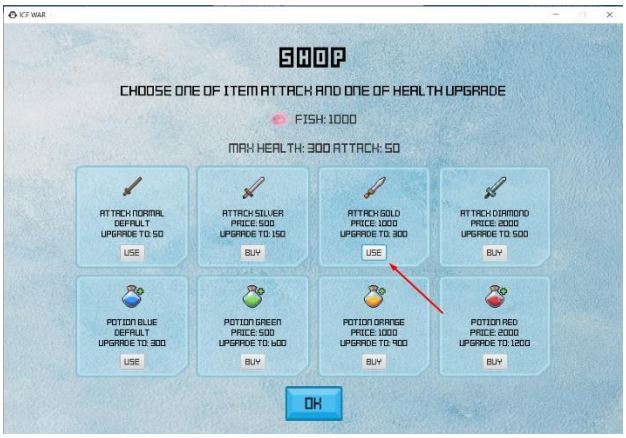
Game Play



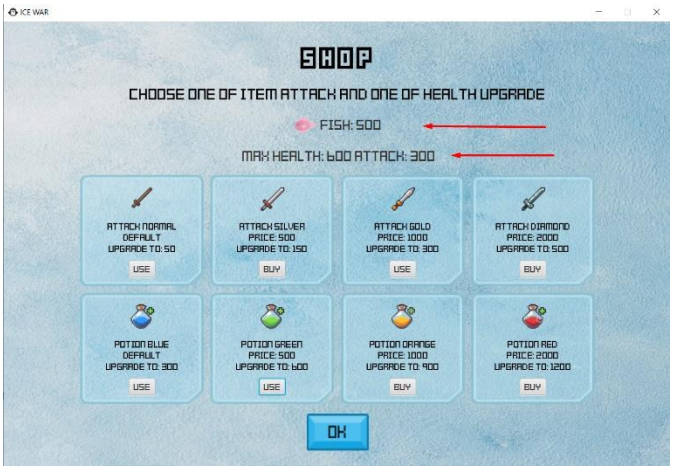
When the player opens the game, the first screen is the main menu screen. There are 4 buttons which are the play button, shop button, how to button and exit button. You can press the play button to start the new round of this game. In each round, the game ends if the player lose all health points or the player finish all level. If you start the next round, it will start at level 1. If the player clicks on the shop button, the player accesses the shop menu. That means the player can collect fish on each round, and when the round ends, the player can go back to the main menu and access the shop for upgrading the character. If you press the exit button, the program will end.

The Shop Menu - Player can buy to upgrading the health potion and weapon attack. When it is already bought, the use button will show up. You can choose one of item attack and one of health upgrade. After choosing, you can look at the update value in the top of screen.

Before Buy



After Buy the button change to use



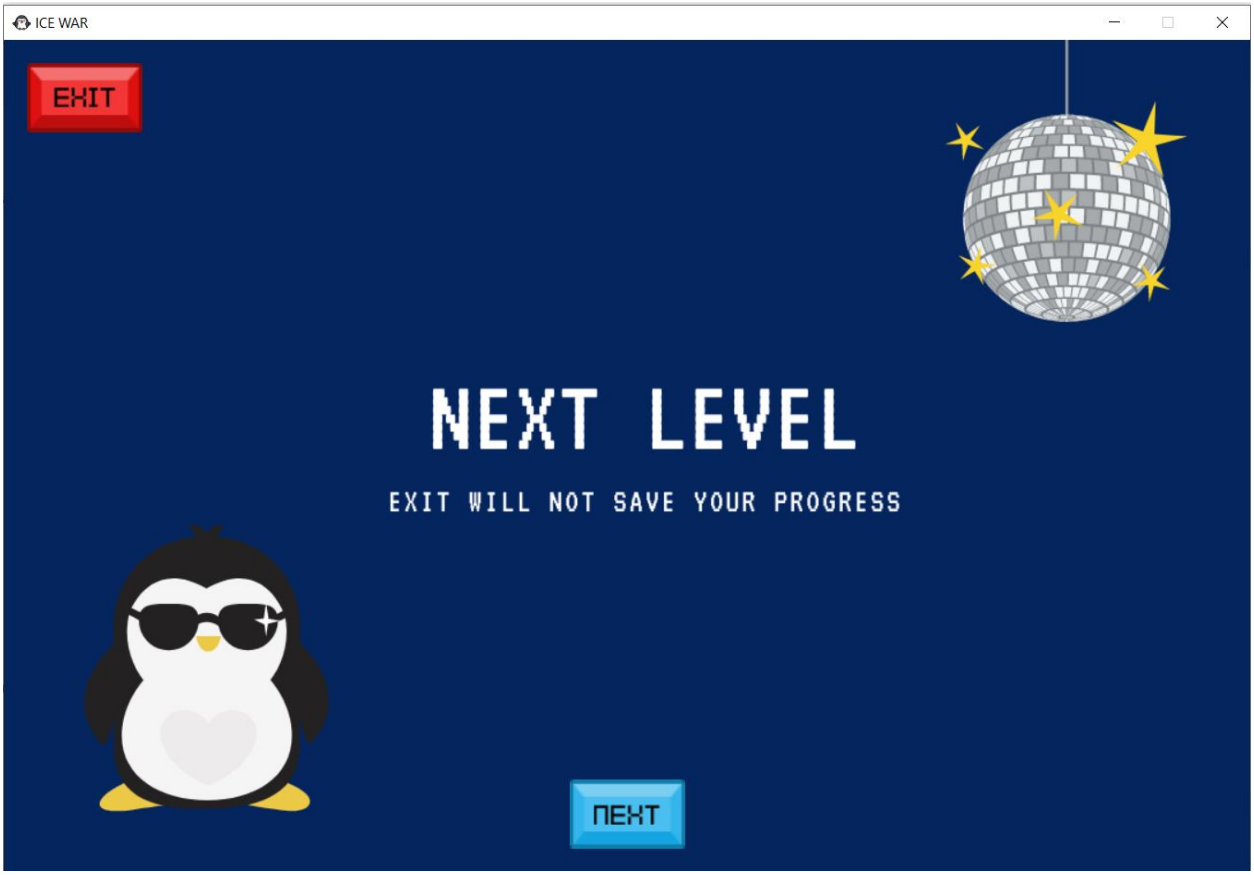
Fish will update after buying success

and the skill will update after you press on button use.

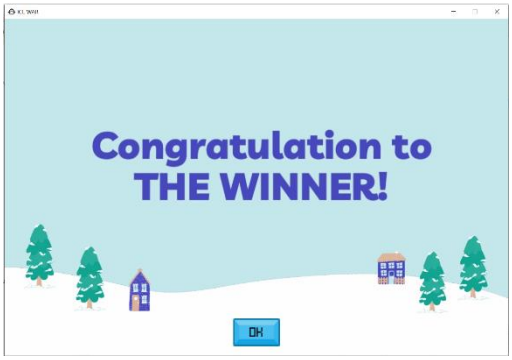
The Manual – Click on how to play button for learning manual of this game

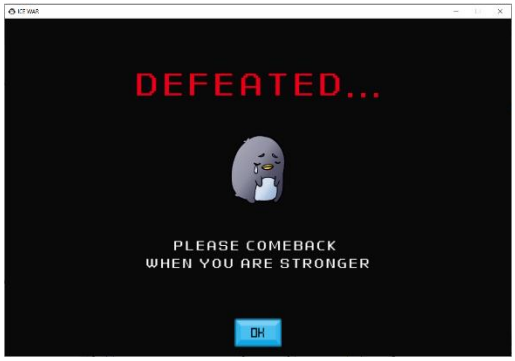
Next Level Scene

When you can pass the level, it will show up this UI, if you click on the next button, it will go to next level, but when you press on the exit button, it didn't save your progress, only save your fish.



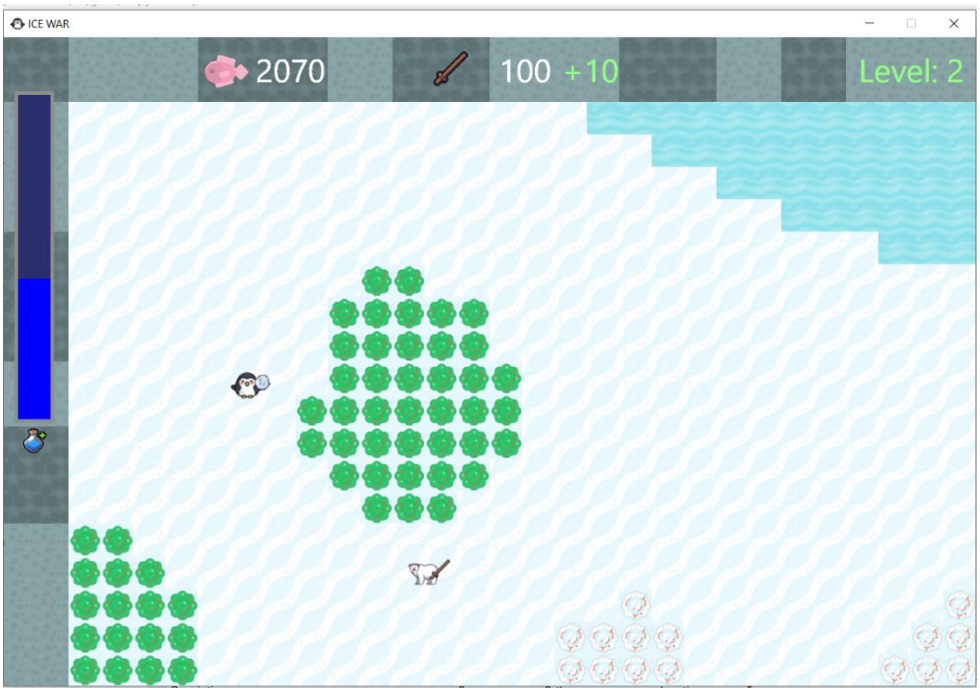
Ending Scene





After the game ended, the player can go back to the main menu.

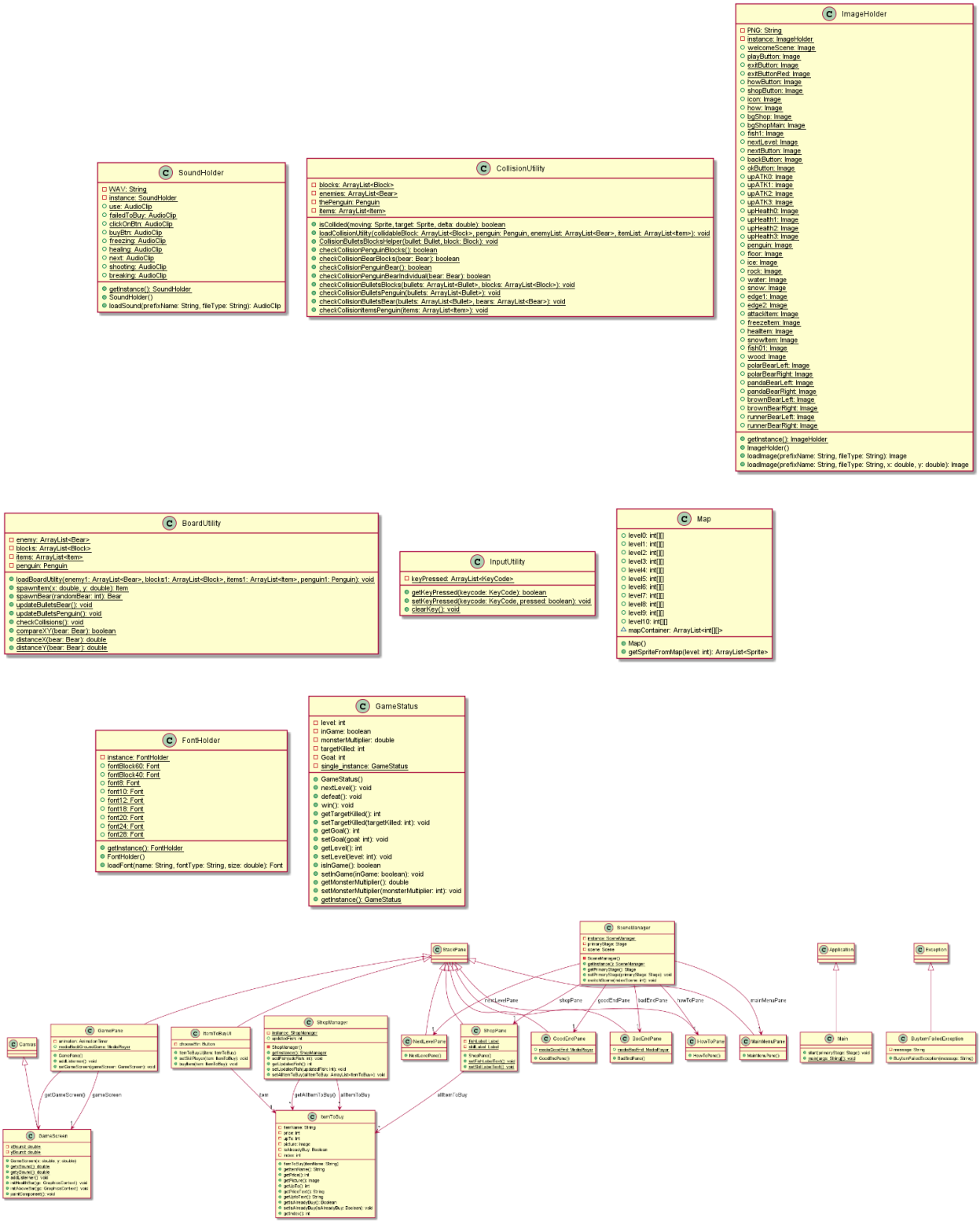
Some examples of game play

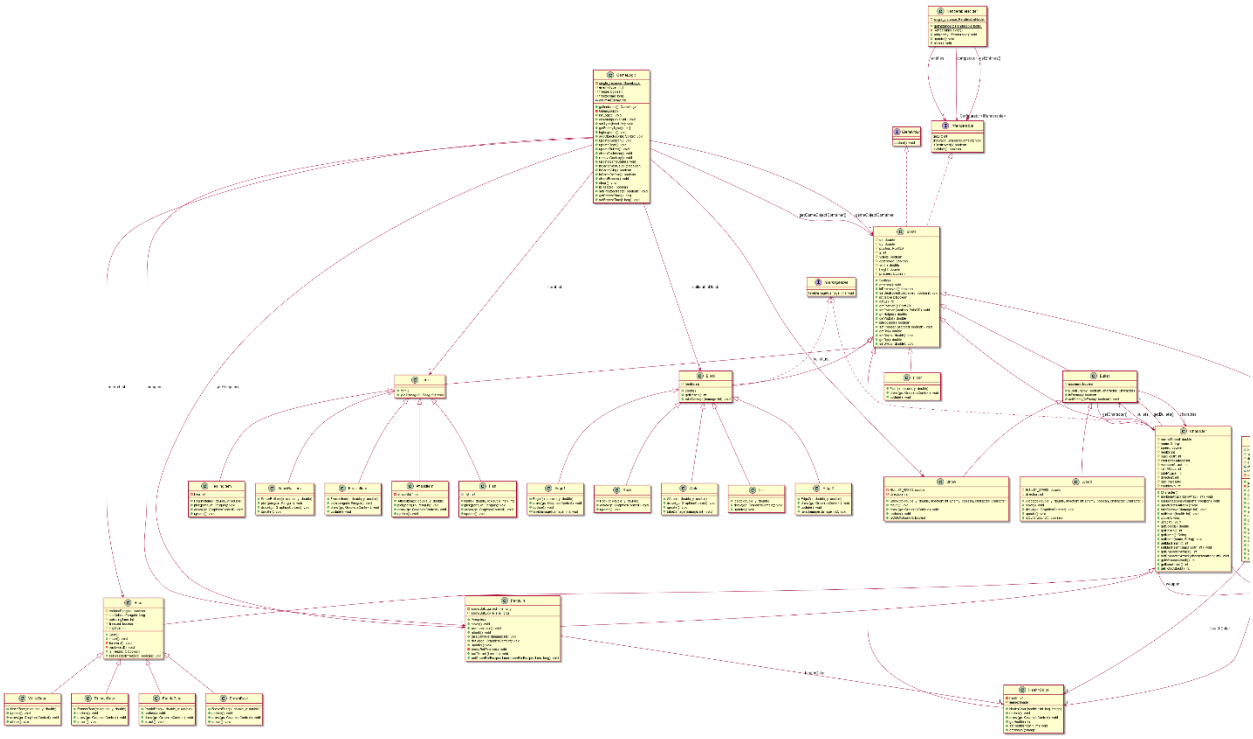


Noted that Access Modifier Notations can be listed below

+ (public), # (protected), - (private) static will be underlined, abstract will be italic

Plant UML


SoundHolder
WAV: String
instance: SoundHolder
use: AudioClip
failedToBuy: AudioClip
clickOnBtn: AudioClip
buyBtn: AudioClip
freezing: AudioClip
healing: AudioClip
next: AudioClip
shooting: AudioClip
breaking: AudioClip
getInstance(): SoundHolder
SoundHolder()
loadSound(prefixName: String, fileType: String): AudioClip
CollisionUtility
blocks: ArrayList<Block>
enemies: ArrayList<Bear>
thePenguin: Penguin
items: ArrayList<Item>
isCollided(moving: Sprite, target: Sprite, delta: double): boolean
loadCollisionUtility(collidableBlock: ArrayList<Block>, penguin: Penguin, enemyList: ArrayList<Bear>, itemList: ArrayList<Item>): void
CollisionBulletsBlocksHelper(bullet: Bullet, block: Block): void
checkCollisionPenguinBlocks(): boolean
checkCollisionBearBlocks(bear: Bear): boolean
checkCollisionPenguinBear(): boolean
checkCollisionPenguinBearIndividual(bear: Bear): boolean
checkCollisionBulletsBlocks(bullets: ArrayList<Bullet>, blocks: ArrayList<Block>): void
checkCollisionBulletsPenguin(bullets: ArrayList<Bullet>): void
checkCollisionBulletsBear(bullets: ArrayList<Bullet>, bears: ArrayList<Bear>): void
checkCollisionItemsPenguin(items: ArrayList<Item>): void
ImageHolder
PNG: String
instance: ImageHolder
welcomeScene: Image
playButton: Image
exitButton: Image
exitButtonPied: Image
howButton: Image
shopButton: Image
icon: Image
how: Image
bgShop: Image
bgShopMain: Image
fish1: Image
nextLevel: Image
nextButton: Image
backButton: Image
okButton: Image
upATK0: Image
upATK1: Image
upATK2: Image
upATK3: Image
upHealth0: Image
upHealth1: Image
upHealth2: Image
upHealth3: Image
penguin: Image
floor: Image
ice: Image
rock: Image
water: Image
snow: Image
edge1: Image
edge2: Image
attackItem: Image
freezeItem: Image
healItem: Image
snowItem: Image
fish01: Image
wood: Image
polarBearLeft: Image
polarBearRight: Image
pandaBearLeft: Image
pandaBearRight: Image
brownBearLeft: Image
brownBearRight: Image
runnerBearLeft: Image
runnerBearRight: Image
getInstance(): ImageHolder
ImageHolder()
loadImage(prefixName: String, fileType: String): Image
loadImage(prefixName: String, fileType: String, x: double, y: double): Image
BoardUtility
enemy: ArrayList<Bear>
blocks: ArrayList<Block>
items: ArrayList<Item>
penguin: Penguin
loadBoardUtility(enemy1: ArrayList<Bear>, blocks1: ArrayList<Block>, items1: ArrayList<Item>, penguin1: Penguin): void
spawnItem(x: double, y: double): Item
spawnBear(randomBear: int): Bear
updateBulletsBear(): void
updateBulletsPenguin(): void
checkCollisions(): void
compareXY(bear: Bear): boolean
distanceX(bear: Bear): double
distanceY(bear: Bear): double
InputUtility
keyPressed: ArrayList<KeyCode>
getKeyPressed(keycode: KeyCode): boolean
setKeyPressed(keycode: KeyCode, pressed: boolean): void
clearKey(): void
Map
level0: int[][]
level1: int[][]
level2: int[][]
level3: int[][]
level4: int[][]
level5: int[][]
level6: int[][]
level7: int[][]
level8: int[][]
level9: int[][]
level10: int[][]
mapContainer: ArrayList<int[][]>
Map()
getSpriteFromMap(level: int): ArrayList<Sprite>
FontHolder
instance: FontHolder
fontBlock60: Font
fontBlock40: Font
font8: Font
font10: Font
font12: Font
font18: Font
font20: Font
font24: Font
font28: Font
getInstance(): FontHolder
FontHolder()
loadFont(name: String, fontType: String, size: double): Font
GameStatus
level: int
inGame: boolean
monsterMultiplier: double
targetKilled: int
Goal: int
single_instance: GameStatus
GameStatus()
nextLevel(): void
defeat(): void
win(): void
getTargetKilled(): int
setTargetKilled(targetKilled: int): void
getGoal(): int
setGoal(goal: int): void
getLevel(): int
setLevel(level: int): void
isInGame(): boolean
setInGame(inGame: boolean): void
getMonsterMultiplier(): double
setMonsterMultiplier(monsterMultiplier: int): void
getInstance(): GameStatus
StackPane
SceneManager
Application
Exception
Canvas
GamePane
FontToBuyUI
ShopManager
NextLevelPane
ShopPane
GoodEndPane
BadEndPane
HowToPane
MainMenuPane
Main
BuyItemFailedException
GameScreen
ItemToBuy
getGameScreen()
gameScreen
getAllItemToBuy()
allItemToBuy
nextLevelPane
shopPane
goodEndPane
badEndPane
howToPane
mainMenuPane



1. Package application

1.1. Class: Main

This class represent the game launcher.

1.1.1. Method

+ void start(Stage primaryStage) throws Exception	Using Scene Manager to set primary stage to this primary stage and then use Scene Manager switch scene, switch these scenes to main menu. After that non resizable and icon.
+ void main(String[] args)	Launch an application

2. Package resource

This package is use for loading all the font, image, sound that we use in the game

2.1. Class: FontHolder

2.1.1. Field

- final FontHolder instance	To create new FontHolder instance and be the only one instance.
+ Font (many variable)	Field to load various font

2.1.2. Constructor

+ FontHolder()	Calling loadFont methos

2.1.3. Method

+ FontHolder getInstance()	Method to load font files
+ Font loadFont(String name, String fontType, double size)	Load font file method

2.2. Class ImageHolder

2.2.1. Fields:

- final ImageHolder instance	To create new ImageHolder instance and be the only one instance.
+ ImageHolder (many variables)	Field to load various image

2.2.2. Construct

+ ImageHolder( )	Calling loadImage method

2.2.3. Method

+ ImageHolder getInstance( )	Method to load font files
+ Image loadImage(String prefixName, String fileType)	Load Image File
+ Image loadImage(String prefixName, String fileType, double x, double y)	Load Image File with size

2.3. Class SoundHolder

2.3.1. Field:

- final SoundHolder instance	To create new SoundHolder instance and be the only one instance.
+ AudioClip (many variables)	Field to load various sound

2.3.2. Construct

+ SoundHolder()	Calling load Sound method

2.3.3. Method

+ SoundHolder getInstance()	Get an instance of SoundHolder
+ AudioCliploadSound(String prefixName, String fileType)	Load sound file method

3. Package gui

This package consists of creation panes and Scene Manager to choose pane to show

3.1. Class: SceneManager

This class is singleton class that represent the manager of the scene. It uses for changing different scenes.

3.1.1. Field

- SceneManager instance	Set SceneManager instant equal to null
- Stage primaryStage	The primaryStage
- Scene scene	Scene
- ShopPane shopPane	To be instance of ShopPane
- MainMenuPane mainMenuPane	To be instance of MainMenuPane
- HowToPane howToPane	To be instance of HowToPane
- GoodEndPane goodEndPane	To be instance of ShopPane
- BadEndPane badEndPane	To be instance of BadEndPane
- NextLevelPane nextLevelPane	To be instance of NextLevelPane

3.1.2. Constructor

- SceneManager( )	To initialize ShopPane, HowToPane, GoodEndPane, BadEndPane, NextLevelPane scene by using MainMenuPane in scene

3.1.3. Method

+ SceneManager getInstance( )	Get instance of SceneManager and if instance is null, initialize SceneManager
+ getPrimaryStage() + setPrimaryStage(Stage primaryStage)	Getter and Setter Primary Stage
+ switchScene(int indexScene)	Using primary stage to set scene by 7 indexScene which are mainMenuPane initialize gamePane, shopPane, howToPane,goodEndPane, badEndPane, nextLevelPane

3.2. Class: MainMenuPane

This class extends from StackPane to create the main menu UI on the pane

3.2.1. Constructor

+ MainMenuPane ( )	Create the MainMenu UI Initialize the background, button, and set on action in each button that call to switchScene by using sceneManager

3.3. Class: GamePane

This class extends from StackPane to create the game pane. Then game screen can use this pane for drawing.

3.3.1. Field

GameScreen gameScreen	The class that drawing on the screen
AnimationTimer animationTimer	Animation timer

3.3.2. Constructor

+ GamePane ( )	Game Status call next level and initialize game screen. Set Focus traversable to be true. Use the animation timer to update and use game status

3.3.3. Method

+ addListener ( )	Set Key Press to true and set key release to false
+ getter and setter GameScreen	

3.4. Class: HowToPane

This class extends from StackPane to create the How To pane UI. It will show after you click on the main menu to see how to play this game.

3.4.1. Constructor

+ HowToPane ( )	Create the How To Pane UI by Initialize manual picture and button to go back to main menu.

3.5. Class: GoodEndPane

This class extends from StackPane to create the good end pane UI. This UI will show if the player can pass all the level and be the winner.

3.5.1. Field

+ MediaPlayer( ) mediaGoodEnd	The media player of the sound

3.5.2. Constructor

+ GoodEndPane ( )	Create the GoodEndPane UI Initialize the media background and button to go to main menu

:

3.6. Class: BadEndPane

This class extends from StackPane to create the bad end pane UI. It will pops up after the main character are dying during the game on.

3.6.1. Field

+ MediaPlayer( ) mediaBadEnd	The media player of the sound

3.6.2. Constructor

+ BadEndPane( )	Create the BadEndPane UI by initialize the background and the button to go to main menu

3.7. Class: NextLevelPane

This class extends from StackPane to create the nexrt level pane UI. It will popsup after the main character is passing the previous level and go to the next level or main menu.

3.7.1. Constructor

+ NextLevelPane ( )	Create the NextLevelPane UI by initialize background and button

3.8. Package: gui.shop

Using to the keep the important class for the shop.

3.8.1. Class: ItemToBuy

3.8.1.1. Field

- String itemName	The item name
- int price	The price of item
- Image picture	The image of item
- Boolean isAlreadyBuy	Check that the item is bought or not
- int index	The index of item

3.8.1.2. Constructor

+ ItemToBuy(String itemName)	To construct the item by using item name

3.8.1.3. Method

+ getter and setter of each field	Getter and setter
+ String getPriceText()	Return the word "price: with the item price but when the price is 0. It's going to change to default
+ String getUptoText() {	Return that the word "Upgrade to : " with the upgrading value of each item

3.8.2. Class: ItemToBuyUI

This package consists of each Item UI Block. It will use to construct in one part of shop pane.

3.8.2.1. Field

- ItemToBuy item	Item from ItemToBuy
- Button chooseButton;	The Button for click (It can be use or but)

3.8.2.2. Constructor

+ ItemToBuyUI(ItemToBuy item)	To construct each Item slot that initialize the background and button to click buy item or use item

3.8.2.3. Method

+ void setSkillPlayer(ItemToBuy item)	Play sound when click on and Player equip weapon or health upgrade that we buy
+ void buyItem(ItemToBuy item) throws BuyItemFailedException	If the fish is more than price, Shop manager will update it and the botton will change text to be use. (If not, throw exception). If you want to use the upgrading item, just press the button again. The last is setFishLabelText that show the update fish on shop pane

3.8.3. Class: ShopPane

This class represent the shop menu

3.8.3.1. Field

- Label fishLabel	The label text of balance fish
- Label skillLabel	The label text of health point and weapon attack that the player use know
- ArrayList<ItemToBuy> allItemToBuy	The item that the player can by store in list

3.8.3.2. Constructor

+ ShopPane()	Initialize background and construct item to buy UI shop manager get allItemTo Buy using the index. Then add all on the pane. Initialize text title, instruction, and button to click back to main menu

3.8.3.3. Method

+ void setFishLabelText()	Set text property value to the fish right now by using shop manager to get the value of fish
+ void setFishLabelText()	Set text property value to the skill which are the weapon attack and health point right now by using shop manager to get the value

3.8.4. Class: ShopManager

This class is singleton class that represent the manager of the shop.

3.8.4.1. Field

- ShopManager instance	Set ShopManagerinstant equal to null
- int updatedFish	The updated fish
- ArrayList<ItemToBuy> allItemToBuy	The all of item, the player can buy

3.8.4.2. Constructor

- ShopManager()	To set the fish and initialize the item to buy to stir in allItemToBuy list

3.8.4.3. Method

+ ShopManagergetInstance( )	Get instance of SceneManager and if instance is null, initialize SceneManager
+ void addFish(int addFish)	Get updated fish when penguin pick fish in the game
+ getter and setter of remainer field	

4. Package Exception

4.1. Class: BuyItemFailedException

4.1.1. Field

- String message	The message show when error occur

4.1.2. Method

+ BuyItemFailedException(String message)	Set message.

5. Package interfaces

5.1. Interface IBehaviour

+ void update( )	A method is used to update position or certain parameters.

5.2. Interface IDamageable

+ void takeDamage(int damage)	A method is used to take damage

6. Package sprite

This package consists of sprite all object for the game

6.1. Package: sprite.base

6.1.1. *Abstract Class:* Sprite

This class is abstract class implement IRenderable, IBehaviour

6.1.1.1. Field

# double dx	Let the value to equal 0.0D. It represents speed in x axis.
# double dy	Initialize to equal 0.0D. It represents speed in y axis.
# Point2D position	Initialize the position point 2D at point (0.0D,0.0D)
# int z	Depth of the sprite
# boolean visible	Check if this sprite is visible
# boolean destroyed	Check if this object is destroyed
# double width	Let the value to equal 36.0D
# double height	Let the value to equal 36.0D

Constructor

+ Sprite()	Initialize sprite by set the visible to true and destroy to false, setZ

6.1.2. Method

+ void destroy( )	Set boolean destroyed to true to indicate that it will be erased soon
+ boolean isDestroyed()	Get value of boolean destroyed
+ boolean isVisible()	Get value of boolean visible
+ int getZ( )	Get value of Z
+ void positionValueCorrection()	dx equal to 0 and dy equal to 0
+ Point2D getPosition()	Get Point2D position
+ void setPosition(Point2D position)	Set Point2D position
+ double getWidth()	Get width of the sprite
+ double getHeight()	Get Height of the sprite
+ getter and setter of remaining field	

6.2. Package sprite.block

This package consists of block for the game

6.2.1. *Abstract Class*: Block

This abstract class extends from sprites and implement IDamagable. It represents all items in the game

6.2.1.1. Field

# int health	The health point of the block

6.2.1.2. Constructor

+ Block()	Instantiates a constructor by calling constructor from super class.

6.2.1.3. Method

+ int getHealth()	Get the health point of the block
+ void takeDamage(int damage)	If the health point after take damage less than zero, set it to zero, else set health point to health minus damage

6.2.2. Class: Edge1

This class extends from Block

6.2.2.1. Constructor

+ Edge1(double x, double y)	Constructor edge1 from super class and set health point, set the position to the new position of point 2D x multiple by getWidth and y multiple by getHeight

6.2.2.2. Method

+ void update()	Don't do any thing
+ void takeDamage(int damage)	Don't do any thing
+ void draw(GraphicsContext gc)	Draw the edge1 block

6.2.3. Class: Edge2

This class extends from Block

6.2.3.1. Constructor

+ Edge2(double x, double y)	Constructor edge2 from super class and set health point, set the position to the new position of point 2D x multiple by getWidth and y multiple by getHeight

6.2.3.2. Method

+ void update()	Don't do any thing
+ void takeDamage(int damage)	Don't do any thing
+ void draw(GraphicsContext gc)	Draw the edge2 block

6.2.4. Class Floor

This class extends from Block

6.2.4.1. Constructor

+ Floor(double x, double y)	Constructor floor from super class, set the position to the new position of point 2D x multiple by getWidth and y multiple by getHeight

6.2.4.2. Method

+ void update()	Don't do any thing
+ void draw(GraphicsContext gc)	Draw the floor block

6.2.5. Class Ice

This class extends from Block

6.2.5.1. Constructor

+ Ice (double x, double y)	Constructor ice from super class and set health point, set the position to the new position of point 2D x multiple by getWidth and y multiple by getHeight

6.2.5.2. Method

+ void update()	If health point is less than 0, call destroy method
+ void draw(GraphicsContext gc)	Draw the ice block but if it is destroyed draw floor block

6.2.6. Class Rock

This class extends from Block

6.2.6.1. Constructor

+ Rock (double x, double y)	Constructor rock from super class and set health point, set the position to the new position of point 2D x multiple by getWidth and y multiple by getHeight

6.2.6.2. Method

+ void update()	If health point is less than 0, call destroy method
+ void draw(GraphicsContext gc)	Draw the rock block but if it is destroyed draw floor block

6.2.7. Class Water

This class extends from Block

6.2.7.1. Constructor

+ Water(double x, double y)	Constructor from super class and set health point, set the position to the new position of point 2D x multiple by getWidth and y multiple by getHeight

6.2.7.2. Method

+ void update()	Don't do any thing
+ void takeDamage(int damage)	Don't do any thing
+ void draw(GraphicsContext gc)	Draw the water block

6.3. Package sprite.equipment

6.3.1. Package sprite.equipment.bullet

6.3.1.1. *Abstract Class*: Bullet

This class is abstract class extends from sprite.

6.3.1.1.1. Field

# boolean isEnemy	Check if this bullet is from the enemy
# Character character	Character that fires this bullet

6.3.1.1.2. Constructor

+ Bullet(boolean Enemy, Character character)	Instantiates a constructor by calling constructor from super class and set the value of z.

6.3.1.1.3. Method

+ Character getCharacter()	Get the character

6.3.1.2. Class: Snow

This class extends from bullet.

6.3.1.2.1. Field

- final double BULLET_SPEED	Speed of the snow
- int direction	Direction of snow

6.3.1.2.2. Constructor

+ Snow (double x, double y, int direction, boolean Enemy, Character character)	Instantiates a constructor by calling constructor from super class and set the bullet speed, width, height, direction and position

6.3.1.2.3. Method

+ void move()	Move in the same direction as character
+ void draw(GraphicsContext gc)	Draw snow
+ void update()	Call method move and check if is out of bound, call destroy
+ boolean isOutOfBound()	Check if it is out of its limited range

6.3.1.3. Class: Wood

This class extends from bullet.

6.3.1.3.1. Field

- final double BULLET_SPEED	Speed of wood
- int direction	Direction of wood

6.3.1.3.2. Constructor

+ Wood(double x, double y, int direction, boolean Enemy, Character character)	Instantiates a constructor by calling constructor from super class and set the bullet speed, width, height, direction, and position

6.3.1.3.3. Method

+ void move()	Move in the same direction as character
+ void draw(GraphicsContext gc)	Draw wood
+ void update()	Call method move and check if is out of bound, call destroy
+ boolean isOutOfBound()	Check if it is out of its limited range

6.3.2. Package sprite.equipment.healthcolor.

6.3.2.1. Class: HealthColor

This class extends from sprite.

6.3.2.1.1. Field

- int health	Max health of penguin
- Image image	Image of health

6.3.2.1.2. Constructor

+ HealthColor(int health1, Image img)	Instantiates all field by input value

6.3.2.1.3. Method

+ void draw(GraphicsContext gc)	Draw this health potion
+ int getHealth()	Get its health
+ void setHealth(int health)	Set its health
+ void update()	Do nothing

6.3.3. Package sprite.equipment.weapon

6.3.3.1. *Abstract Class*: Weapon

This class is abstract class extends from sprite.

6.3.3.1.1. Field

# int weaponAttack	Attack that weapon will give to penguin

6.3.3.1.2. Constructor

+ Weapon( )	Instantiates a constructor by calling constructor from super class and set the width, height andvalue of z.

6.3.3.1.3. Method

+ int getWeaponAttack()	Return weaponAttack
+ void setWeaponAttack (int weaponAttack)	Set weaponAttack
+ void update()	Do nothing
+ *void update(double x, double y)*	Abstract class to update position of weapon

6.3.3.2. Class: Snowball

This class extends from weapon.

6.3.3.2.1. Field

- Image image	Image of this class

6.3.3.2.2. Constructor

+ Snowball(Weapon oldWeapon)	Instantiates a constructor by calling constructor from super class and set the weapon attack and image

6.3.3.2.3. Method

+ void draw(GraphicsContext gc)	Draw the snowball
+ void update()	Call method move and check if is out of bound, call destroy

6.3.3.3. Class: Stick

This class extends from weapon.

6.3.3.3.1. Field

- Image image	Image of this class

6.3.3.3.2. Constructor

+ Stick(int weaponAttack1, Image img)	Instantiates a constructor by calling constructor from super class and set the weapon attack and image

6.3.3.3.3. Method

+ void draw(GraphicsContext gc)	Draw the stick
+ void update()	Update its position

6.4. Package sprite.character

6.4.1.1. *Abstract Class*: Character

This class is abstract class extends from sprite implements IDamageable.

6.4.1.1.1. Field

# double normalSpeed	Overall speed of all character
# int health	Speed of character
# int characterAttack	Character's base attack
# int weaponAttack	Character's weapon attack
# int itemAttack	Character's item attack
# int totalAttack	Total attack of character
# Weapon weapon	Weapon of character
# int direction	Character's direction
# ArrayList<Bullet> bullets	Bullets from this character
# long lastFired	Last time that this character fire bullet
# int cooldown	Cooldown to slow down time to fire next bullet

6.4.1.1.2. Constructor

+ Character( )	Instantiates a constructor by calling constructor from super class and set the value of z.

6.4.1.1.3. Method

+ Character getCharacter()	Get the character
+ void addItemAttack(int itemAttack)	Increase character's item attack by itemAttack
+ void equipWeapon(Weapon weapon1)	Character equips new weapon
+ void updateTotalAttack()	Summation of all attack
+ void takeDamage(int damage)	Take damage if health less than 0 set to 0 else set to current health minus damage
+ *void move()*	Abstract class to move
+ *void attack()*	Abstract class to attack
+ void setHealth(int health)	If the health more than max health set to equal max health but if it is less than zero, set it to zero
+ getter and setter for remaining field	

6.4.1.2. Class: Penguin

This class extends from character implements Inputable.

6.4.1.2.1. Field

- HealthColor healthColor	Penguin's health Color
- long snowBallEquipedTime	Time when penguin last equip snowball
- long snowBallExpireTime	Time limits that penguin can use snowball before it expire

6.4.1.2.2. Constructor

+ Penguin()	Instantiates a constructor by calling constructor from super class and set the all of value from player and call update total attack

6.4.1.2.3. Method

+ void move()	Move in different direction according to input from keyboard
+ void processInput()	Call move() and attack()
+ void attack()	Attack according to input from keyboard
+ void takeDamage(int damage)	Take damage same as character but also take damage to player
+ void draw(GraphicsContext gc)	Draw penguin

+ void update()	Call method move and check if is out of bound, call destroy
+ void snowBallExpired()	Equip stick if current time exceed expired time
+ void buffAttack(int item)	Increase itemAttack by item
+ void setSnowBallEquipedTime(long snowBallEquipedTime)	Set snowBallEquipedTime when it is equiped

6.4.1.3. Package: sprites.character.bear

6.4.1.3.1. Abstract Class: Bear

This class extends from character.

6.4.1.3.2. Field

# boolean collidedPenguin	Check if this collides with penguin
# long lastCollidedPenguin	Indicate time that this collide with penguin
# int cofusingTime	Time that bear will be confused
# boolean freezed	Chrck if it is in freezing state
# int imgSide	Reference value to know which direction bear will be draw

6.4.1.3.3. Constructor

+ Bear()	Instantiates a constructor by calling constructor from super class

6.4.1.3.4. Method

+ void move()	Move according to logic
+ void forward()	Move forward
+ void backward()	Move backward
+ boolean isFreezed()	Get if it in freezing state
+ void setFreezed(boolean freezed)	Set this freezing state by freezed

6.4.1.3.5. Class: White Bear

This class extends from bear.

6.4.1.3.5.1. Constructor

+ WhiteBear(double x, double y)	Instantiates a constructor by calling constructor from super class and set the value

6.4.1.3.5.2. Method

+ void update()	Set this position, fire bullet if it is not in cooldown, and freeze it if in freezing state
+ void draw(GraphicsContext gc)	Draw the white bear
+ void attack()	Attack with corresponding weapon and add bullet to its bullets

6.4.1.3.6. Class: Brown Bear

This class extends from bear.

6.4.1.3.6.1. Constructor

+ BrownBear(double x, double y)	Instantiates a constructor by calling constructor from super class and set the value

6.4.1.3.6.2. Method

+ void update()	Set this position, fire bullet if it is not in cooldown, and freeze it if in freezing state
+ void draw(GraphicsContext gc)	Draw the Brown bear
+ void attack()	Attack with corresponding weapon and add bullet to its bullets

6.4.1.3.7. Class: Panda Bear

This class extends from bear.

6.4.1.3.7.1. Constructor

+ PandaBear(double x, double y)	Instantiates a constructor by calling constructor from super class and set the value

6.4.1.3.7.2. Method

+ void update()	Set this position, fire bullet if it is not in cooldown, and freeze it if in freezing state
+ void draw(GraphicsContext gc)	Draw the panda bear
+ void attack()	Attack with corresponding weapon and add bullet to its bullets

6.4.1.3.8. Class: Runner Bear

This class extends from bear.

6.4.1.3.8.1. Constructor

+ RunnerBear(double x, double y)	Instantiates a constructor by calling constructor from super class and set the value

6.4.1.3.8.2. Method

+ void update()	Set this position, fire bullet if it is not in cooldown, and freeze it if in freezing state
+ void draw(GraphicsContext gc)	Draw the runner bear
+ void attack()	Attack with corresponding weapon and add bullet to its bullets

6.5. Package sprite.item

This package consists of item for the game

6.5.1. *Abstract Class*: Item

This abstract class extends from sprites.

6.5.1.1. Constructor

+ Block()	Instantiates a constructor by calling constructor from super class and set the value of z.

6.5.1.2. Method

+ void pick (Penguin penguin)	Abstract method picking item

6.5.2. Class: Fish

This class extends from Item

6.5.2.1. Constructor

+ Fish(double x, double y, int fish)	Constructor Fish from super class, set the position and random amount of fish that get after pick this

6.5.2.2. Method

+ void update()	Don't do any thing
+ void pick(Penguin penguin)	If player pick the fish, it will increase fish
+ void draw(GraphicsContext gc)	Draw the Fish item

6.5.3. Class: HealingItem

This class extends from item

6.5.3.1. Field

- int Heal	Amount that this will heal penguin

6.5.3.2. Constructor

+ HealingItem(double x, double y)	Constructor Healing Item from super class and set heal point, set the position.

6.5.3.3. Method

+ void update()	Don't do any thing
+ void pick(Penguin penguin)	If penguin pick, add healing point, but do not exceed maximum health
+ void draw(GraphicsContext gc)	Draw the healing item

6.5.4. Class FreezeItem

This class extends from item

6.5.4.1. Constructor

+ FreezeItem(double x, double y)	Constructor floor from super class, set the position

6.5.4.2. Method

+ void update()	Don't do any thing
+ void draw(GraphicsContext gc)	Draw the freeze item
+ void void pick(Penguin penguin)	Stop movement of all enemy

6.5.5. Class AttackItem

This class extends from item

6.5.5.1. Field

- int attackBuff	Increase penguin's item attack by this

6.5.5.2. Constructor

+ AttackItem(double x, double y)	Constructor attack item from super class and set the position

6.5.5.3. Method

+ void update()	Do nothing
+ void pick(Penguin penguin)	Increase penguin's item attack by this
+ void draw(GraphicsContext gc)	Draw the attack item

6.5.6. Class SnowBallItem

This class extends from item

6.5.6.1. Constructor

+ AttackItem(double x, double y)	Constructor snowball item from super class and set the position

6.5.6.2. Method

+ void update()	Do nothing
+ void pick(Penguin penguin)	Equip snow ball
+ void draw(GraphicsContext gc)	Draw the attack item

7. Package Render

7.1. Interface: IRenderable

7.1.1. Method

+ int getZ()	Get the Z-axis(depth) number
+ void draw(GraphicsContext gc)	Draw object, depends on each object
+ boolean isDestroyed()	Check if an object visible
+ boolean isVisible()	Check if an object is destroyed

7.2. Class: RenderableHolder

This class is singleton class.

7.2.1. Field

- RenderableHolder single_instance	Set RenderableHolder single_instant equal to null
- List<IRenderable> entities	List that contains object that will be drawn
- Comparator<IRenderable> comparator	Set order of drawing entities by compare z value

7.2.2. Constructor

- RenderableHolder ( )	Initialize entities and comparatpr

7.2.3. Method

+ RenderableHolder getInstance( )	Get instance of RenderableHolder and if instance is null, initialize RenderableHolder()
+ void add(IRenderable entity)	Add entity to List<IRenderable> entities and sort them
+ void update()	Remove objects that are destroyed
+ List<IRenderable> getEntities()	Get List<IRenderable> entities
+ void clear( )	Set single_instance to null to clear all object in List<IRenderable> entities

8. Package game

8.1. Class: GameScreen

This class extends from canvas

8.1.1. Field

- double xBound	Bound of screen in x axis
- double yBound	Bound of screen in y axis

8.1.2. Constructor

+ GameScreen(double x, double y)	The main method for this application. It initialize xBound and yBound, and call method addListerner()

8.1.3. Method

+ void paintComponent()	Draw every object in RendernableHolder if it is not destroyed and is visible
+ void addListerner()	set on input key release and set on key press by calling getKeyPressed(KeyCode keycode) or setKeyPressed(KeyCode keycode,boolean pressed)
+ void initHealthBar(GraphicsContext gc)	Draw in canvas to show the health point
+ void abovBar(GraphicsContext gc)	Draw in canvas to show the stat
+ getters and setters xBound and getBound	

8.2. Class: GameLogic

This class is singleton class that represent the logic of each game.

8.2.1. Field

- GameLogic single_instance	Set GameLogic single_instant equal to null
- ArrayList<Block> collidableBlock	Contains block that can be hit (Block class)
- ArrayList<Snow> bulletList	Contains bullets (Bullet class) of every character (Character class)
- ArrayList<Bear> enemyList	Contains every enemy (Bear class)
- ArrayList<Item> itemList	Contain every item (Item class) that is droped in map
- ArrayList<Sprite> gameObjectContainer	Contain every Sprite that have to update their logic
- Penguin penguin	Player will control this
- int [ ] enemyType	Contains number of enemies in the level
- boolean freeze	Indicate the freezing state of Bear
- long freezeTime	Time that FreezeItem last picked
- int countAllEnemy	Count number of enemy that have been appeared in the map

8.2.2. Constructor

- GameLogic( )	Call init logic method

8.2.3. Method

+ GameLogic getInstance( )	Get instance of GameLogic and if instance is null, initialize GameLogic
+ void initLogic()	Initialize all field, CollisionUtility and BoardUtility, and call addObject(penguin)
+ void drawMap(int level)	Initialize new Map and call map.getSpriteFromMap(level) to add its value to suitable ArrayList by calling addObject
+ void setType(int level)	Set enemyType to correspond with level
+ void logicUpdate()	Update all logic
+ void addObject(Sprite sprite)	add its value to suitable ArrayList and RenderableHolder
+ void updateWeapon()	Update position of all weapons
+ void updateBear()	Update logic of all bears
+ void updateBullets()	Update logic of all bullets
+ void checkCollisions()	Update and check if anything is collided
+ void removeDestroy()	Removeof all destroyed Sprite in all ArrayList
+ void updateGameState()	Update position of all weapons
+ boolean isGameNextLevel()	Check if player clear the level, clear all data, and move to NextLevelScene

+ boolean isGameWin()	Check if player win the game , clear all data, and move to GoodEndingScene
+ boolean isGameDefeat()	Check if player is defeated, clear all data, and move to BadEndingScene
+ void checkFreeze()	Check if freeze is true or false and update value of position of bear accordingly
+ void clear()	Call initLogic() to clear all value and also clear RenerdableHolder
+ List<Sprite> getGameObjectContainer()	Get gameObjectContainer
+ Penguin getPenguin()	Get penguin
+ boolean isFreeze()	Check if is in freeze state
+ void setFreeze(boolean freeze)	Set freeze state
+ long getFreezeTime()	Get time that it lasted freeze
+ void setFreezeTime(long l)	Set time that it lasted freeze

8.3. Class: GameStatus

This class is singleton class that represent the status of each game.

8.3.1. Field

- GameStatus single_instance	Set GameStatus single_instant equal to null
- int level	The level of game
- boolean inGame	The Boolean chek if is in a game
- double monsterMultiplier	Scale that indicate how monster become stronger in later state
- int targetKilled	Current amount of monsters that are killed
- int Goal	Number of monsters that need to kill to clear the current level

8.3.2. Constructor

+ GameStatus()	Set the level, targaetKilled to zero. And set goal monsterMultiplier

8.3.3. Method

+ GameStatus getInstance( )	Get instance of GameStatus and if instance is null, initialize GameStatus
+ void nextLevel()	Set to fields in this class to appropriate values, increase level and initialize new level
+ void defeat()	Set to fields in this class to appropriate values
+ void win()	Set to fields in this class to appropriate values
+ getter and setter for the remainder field	

8.4. Class: Map

This class represent all items in the game

8.4.1. Field

+ int[][] level0 (to 10 there are many level)	All level contains int that indicate type of block in map
- ArrayList<int[][]> mapContainer	Contain all level map

8.4.2. Constructor

+ Map()	Add the map of each level to mapContainer and enemy in each Level to typeEnemy

8.4.3. Method

+ ArrayList<Sprite> getSpriteFromMap(int level)	Return ArrayList that contain Sprite that change according to indicators in level

8.5. Class: Player

This class is singleton class that represent the player who play the game.

8.5.1. Field

- Player single_instance	Set Player single_instant equal to null
- int baseAttack	Base attack of character penguin
- Weapon weapon	Weapon of character penguin
- int weaponAttack	Weapon attack of character penguin
- HealthColor healthColor	HealthColor of character penguin
- int currentHealth	Current health of character penguin
- int maxHealth;	Maximum health of character penguin
- int totalAttack	Total attack of character penguin

8.5.2. Constructor

- Player()	Set the value of base attack and item attack. Call the method equip weapon and Health color, then set the value of weapon attack and max health point, current health point, then call method update total attack

8.5.3. Method

+ Player getInstance( )	Get instance of Player and if instance is null, initialize Player and its all fields
+ int getWeaponAttack()	Get weapon attack of player

+ int getTotalAttack()	Get total weapon attack of player
+ void equipWeapon(Weapon weapon1)	Set weapon of player to this weapon and change total attack accordingly
+ void setWeaponAttack(int weaponAttack)	Set weapon attack of player
+ void equipHealthColor(HealthColor healthColor1)	Set HealthColor of player to this HealthColor and change health accordingly
+ void addItemAttack(int itemAttackBuff)	Increase item attack and total attack accordingly
+ void updateTotalAttack()	Return summation of base attack, weapon attack and item attack
+ getter and setter of the remainder field	

8.6. Package: utility

This package represents the utility of this game

8.6.1. Class: InputUtility

8.6.1.1. Field

- ArrayList<KeyCode> keyPressed	ArrayList that contain inputs that will be processed to command character action

8.6.1.2. Method

+ boolean getKeyPressed(KeyCode keycode)	Get ArrayList keyPressed
+ void setKeyPressed(KeyCode keycode,boolean pressed)	Set KeyPressed if keyboard is pressed and remove if it is not pressed

8.6.2. Class: BoardUtility

This class represent all items in the game

8.6.2.1. Field

- ArrayList<Bear> enemy	Contains bears
- ArrayList<Block> blocks	Contains collideable blocks
- ArrayList<Item> items	Contains items
- Penguin penguin	Current penguin

8.6.2.2. Method

+ void loadBoardUtility(ArrayList<Bear> enemy1, ArrayList<Block> blocks1	Load data enemy, blocks, items and penguin.

,ArrayList<Item> items1, Penguin penguin1)	
+ Item spawnItem(double x, double y)	Add new item with random possibility to RenderableHolder and GameLogic
+ Bear spawnBear(int randomBear)	Add new bear to RenderableHolder and GameLogic
+ void updateBulletsBear()	Update logic of all bullet of all bear
+ updateBulletsPenguin()	Update logic of all bullet of penguin
+ void checkCollisions()	Check and update logic of all possible way of collision
+ boolean compareXY(Bear bear)	Compare distance of x axis and y axis of this bear and penguin
+ double distanceX(Bear bear)	Get distance of x axis of this bear and penguin
+ double distanceY(Bear bear)	Get distance of y axis of this bear and penguin

8.6.3. Class: CollisionUtility

This is the class for checking collision.

8.6.3.1. Field

- ArrayList<Block> blocks	Contain collidable block
- ArrayList<Bear> enemies	Contain bears
- ArrayList<Item> items	Contain items
- Penguin thePenguin	Current penguin

8.6.3.2. Method

+ boolean isCollided(Sprite moving, Sprite target, double delta)	Check if two Sprite are collided with specific degree
+ void loadCollisionUtility(ArrayList<Block> collidableBlock, Penguin penguin, ArrayList<Bear> enemyList, ArrayList<Item> itemList)	Load data enemy, blocks, items and penguin.
+ void CollisionBulletsBlocksHelper (Bullet bullet, Block block)	Update logic of collision of collidable blocks and bullets
+ boolean checkCollisionPenguinBlocks()	Check if penguin collide with any blocks
+ boolean checkCollisionBearBlocks(Bear bear)	Check if individual bear collide with any blocks
+ boolean checkCollisionPenguinBear()	Check if penguin collide with any bears

+ boolean checkCollisionPenguinBearIndividual (Bear bear)	Check if penguin collide with this bear
+ void checkCollisionBulletsBlocks(ArrayList<Bullet> bullets, ArrayList<Block> blocks)	Update logic of collision of any bullets and any blocks
+ void checkCollisionBulletsPenguin(ArrayList<Bullet> bullets)	Update logic of collision of any bullets and penguins
+ void checkCollisionBulletsBear(ArrayList<Bullet> bullets, ArrayList<Bear> bears)	Update logic of collision of any bullets and any bears
+ void checkCollisionItemsPenguin (ArrayList<Item> items)	Update logic of collision of any item and penguin