Jie Li
Wingman Game
Csc413

Observerable

<Interface>
Observer

Game1942WithObserver

thread: Thread
sea: Image
myPlane2: Image
myPlane1: Image
island1: Image
island2: Image
island3: Image
bossImg: Image
bottom: Image
bullets: LlinkedList<Bullet>
enemyBullets: LInkedList<EnemyBullet>
enemies; LinkedList<Enemy>
explosions: LinkedList<Explosion>
boss: Boss
bimg; BufferedImage
g2: Graphics2D
move: int
speed: int
generator: Random
I3: Island
I2: Island
I1; Island
m1: MyPlane
m2: MyPlane
h: int
w: int
gameEvents: GameEvents
t: TimeLine
controller: Controller
bm: backGroundMusic

+ init():
+ drawBackGroundWithTileImage():
+ drawDemo():
+ paint(Graphics: g):
+ start():
+ playBackgroundMusic():
+ run():
+ main(String[] argv):

Function: Update and draw the frame continuously as long as the thread is not interrupted.

GameEvents

enemyEvent: boolean
enemyBulletEvent: boolean
enemy_with_plane1: boolean
enemy_with_plane2: boolean
bullet_with_plane1: boolean
bullet_with_plane2: boolean
enemy_with_myBullet: boolean
bossEvent: boolean
boss_with_myBullet: boolean
boss_with_plane1: boolean
boss_with_plane2: boolean
bossDie: boolean
keys: LlinkedList<Integer>

+ GameEvents()
+ keyEvent(LinkedList keys):
+ enemy_MyPlane_Event(int planeNumber):
+ enemy_MyBullet_Event()
+ enemyBulletEvent(int planeNumber):
+ boss_MyPlane_Event(int planeNumber):
+ boss_MyBullet_Event(int planeNumber):
+ boss_is_dead()

Function: This class accepts all events that MyPlane interests in, such as enemy event, enemy's bullet event, keyboard event, boss event, and MyPlane's bullet event. As an observerable, GameEvents sets changes and notifies MyPlane by invoking update() method in MyPlane.

MyPlane

img: Image
width: int
height: int
y: int
x: int
speed: int
bbox: Rectangle
boom: boolean
can_shoot : int
planeNumber: int
health: int
score: int
bullets: LinkedList<Bullet>

+ MyPlane(Image img, int x, int y, int speed, int planeNumber, LinkedList<Bullet> bullets, LinkedList<Explosion> explosions)
+ collision(int x, int y, int w, int h): boolean
+ update(Observable obj, Object arg):
+ draw(ImageObserver obs, Graphics2D: g2):

Function: As an Observer, MyPlane updates its state and behavior according to different events that it accepts from GameEvents. It also provides collision method for enemy plane, enemy's bullet and boss to check collison with it.

ExplosionMusic

MGP: AudioPlayer
BGM: AudioStream

+ ExplosionMusic(String filename)
+ play():

Function: ExplosionMusic is used to create music whenever the explosion happens.

Explosion

img: Image
y: int
x: int

+ Explosion(Image img, int x, int y)
+ draw(ImageObserver obs, Graphics2D: g2):

Function: Explosion object is created by Explosions to create a group of 6 explosions.

backGroundMusic

sequencer: Sequencer
is: InputStream

+ backGroundMusic(String filename)
+ play():
+ end():

Function: backGroundMusic is used to play the background music looply.

Explosions

explosions: LinkedList<Explosion>
explosionImgs: Image[]
y: int
x: int

+ Explosions(int type, int x, int y)
+ getExplosions(): LinkedList<Explosion>:

Function: Explosions is used to create a group of six explosion. getExplosions() method is invoked to get a group of explosion.

Island

img: Image
y: int
x: int
speed: int
gen: Random

+ Island(Image img, int x, int y, int speed, Random gen)
+ update():
+ draw(ImaegObserver obs, Graphics2D: g2):

Function: Island is used to create islands. Whenver the island is out of screen, reset the x coordinate.

Boss

img: Images
x: int
sizeX: int
y: int
speed: int
sizeY: int
hits: int
frames: int
show: boolean
m2: MyPlane
m1: MyPlane
gameEvents: GameEvents

+ Boss(Image img, int speed, int x, int y, MyPlane m1, MyPlane m2, GameEvents gameEvents, LinkedList<EnemyBullet> enemyBullets, LinkedList<Bullet> bullets, LinkedList<Explosion> explosions)
+ update():
+ draw(ImageObserver obs, Graphics2D g2):

Function: Boss class is used to create boss object. In its update methods, it updates its position by each frame, it fires enemy's bullets according to the frame number, it also checks for collision with MyPlane and MyPlane's bullet and creates explosion for MyPlane and MyPlane's bullet. Finally, it reports every event to GameEvents by invoking corresponding methods in GameEvents.

Controller

m1: MyPlane
m2: MyPlane
bottom: Image

+ Controller(MyPlane m1, MyPlane m2, Image bottom)
+ draw(ImageObserver obs, Graphics2D g2):

Function: Controller class is used to draw the bottom image, planes' health and scores.

KeyAdapter

KeyControl

gameEvents: GameEvents
keys: LinkedList<Integer>

+ KeyControl(GameEvents gameEvents)
+ keyPressed(KeyEvent e):
+ keyReleased(KeyEvent e):

Function: KeyControl is used to listen to the keyboard events and report the events to GameEvents by invoking GmaeEvents's corresponding methods. Both keyPressed and keyReleased method is implemented to support multiple-key pressed and released event.

TimeLine

frame: int
enemies: LinkedList<Enemy>
explosions: LinkedList<Explosion>
bullets: LinkedList<Bullet>
boss: Boss
enemy1Image: Image
enemy2Image: Image
enemy3Image: Image
enemy4Image: Image
m2: MyPlane
m1: MyPlane
gameEvents: GameEvents
enemyBullets: LinkedList<EnemyBullet>

+ TimeLine(LinkedList<Enemy> enemies, Boss boss, MyPlane m1, MyPlane m2, GameEvents gameEvents, LinkedList<Bullet> bullets, LinkedList<Explosion> explosions, LinkedList<EnemyBullet> enemyBullets)
+ update()

Function: TimeLine is used to create a wave of enemies according to the frame number. An instance of TimeLine should be declared inside the init() method. Update() method should be invoked in every frame.

Enemy

img: Image
sizeY: int
y: int
sizeX: int
x: int
speed: int
enemyType: int
show: boolean
m1: MyPlane
m2: MyPlane
gameEvents: GameEvents
bullets: LinkedList<Bullet>
enemyBullets: LinkedList<EnemyBullet>
explosions: LinkedList<Explosion>

+ Enemy(Image img, int speed, int x, int y, int enemyType, MyPlane m1, MyPlane m2, GameEvents gameEvents, LinkedList<Bullet> bullets, LinkedList<Explosion> explosions, LinkedList<EnemyBullet> enemyBullets)
+ update():
+draw(ImageObserver obs, Graphics2D g2):

Function: Enemy class is used to created different types of enemis. In update method, it updates its position and checks for collison with MyPlane MyPlane's bullet, and it is also responsible for out of screen event and creating enemy's bullet. Finally, it reports every event to GameEvents by invoking corresponding methods in GameEvents.

EnemyBullet

img: Image
type: int
Enemy_x: int
x: int
sizeX: int
y: int
speed: int
sizeY: int
show: boolean
m2: MyPlane
m1: MyPlane
gameEvents: GameEvents
explosions: LinkedList<Explosion>

+ EnemyBullet(Image img, int type, int speed, int x, int y, MyPlane m1, MyPlane m2, LinkedList<Explosion> explosions, GameEvents gameEvents)
+ update():
+ draw(ImageObserver obs, Graphics2D g2):

Function: EnmeyBullet object is created by Enemy. In update() method, it updates its position, it checks for collision with MyPlane and each of enemy'sbullets and create explosion, and it is also responsible for out of screen event. Finally, it reports every event to GameEvents by invoking corresponding methods in GameEvents.

Bullet

img: Image
y: int
x: int
speed: int
sizeX: int
sizeY: int
show: boolean
bulletBox: Rectangle

+ Bullet(Image img, int speed, int x, int y)
+ collision(int x, int y, int w, int h): boolean
+ update():
+ draw(ImageObserver obs, Graphics2D g2)

Function: Bullet is used to create the bullet for MyPlane use. It provide collision method for enmey and boss to check collision with it. update() method is called every frame to update each bullet's state and behavior according to different events it meets.

Why do I make changes to the class diagram?

The biggest reason for me to change the original class diagram is that I was not clear about which classes should be "Observable" and which classes should be "Observer". In my original diagram, GameEvents, MyBullet, Enemy and EnemyBullet classes are "Observable"; MyPlane and Enemy classes are "Observer". After I figured out every kinds of events can be reported to GameEvents, I decided that GameEvents is the only Observerable, so there are a large amount of fields and methods are added to the GameEvents class.

After I figured out the relationship among all classes, I know that Bullet class has quiet different signatures from EnemyBullet.I choose to let EnemyBullet to report all events to GameEvents including its collision with MyBullet, so MyBullet only needs to have a collision method to let EnemyBullet check with without reportin any events, which is similar to MyPlane. Thus, I think that it would be better way to simply split these 2 classes without inheriting from an Abstract class -- Weapon. I also changed MyBullet class name into Bullet, which is for reuse in Tank game.

Third, it is a little late for me to realize that everything should be updated and drawn in the method drawDemo() in the Game1942WithObserver class, which means every object shares the same list objects -- LinkedList<Enemy>, LinkedList<MyBullet>, LinkedList<EnemyBullet> and LinkedList<Explosion>. References to some of those lists need to be passed to the class constructor, so I changed almost every class's instructor.

Forth, I add a list of Integers in KeyControl class to store the key codes corresponding to the keys that the user pressed, and remove them whever the user released keys. The reason for doing that is because planes don't repond correctly when I press multiple keys together, which doesn't provide the fluent user experience. Thus, I add some work to the KeyControl class, although it is not efficient.

Last, I add TimeLine class, explosionMusic class, and backgroundMusic class. I originally was not sure about how to set TimeLine, but after taking lecture, I know how to set it. I split the way to play music into 2 classes -- explosionMusic class and backgroundMusic class, and the reason is that the music resources have different types; one is wav and another one is mid. I don't know how to play them by using only one class, so I just write two class for them.