

Wingman



Wingman

- Collision Handling (airplane vs. airplane, airplane vs. bullets)
- Timeline handling (show diverse types of enemy planes)
- Explosion effect and explosion sound (small and large)
- Bullets shooting in diagonal direction (enemy and player)
- Bullets shooting towards player
- Power up weapon pick up (show icon and remaining time)
- Score board at the end of game (file read and write)
- Health bar and spare life
- Background music
- resizable
- Support for 2 players with multiple lives
- Smooth performance

Tank



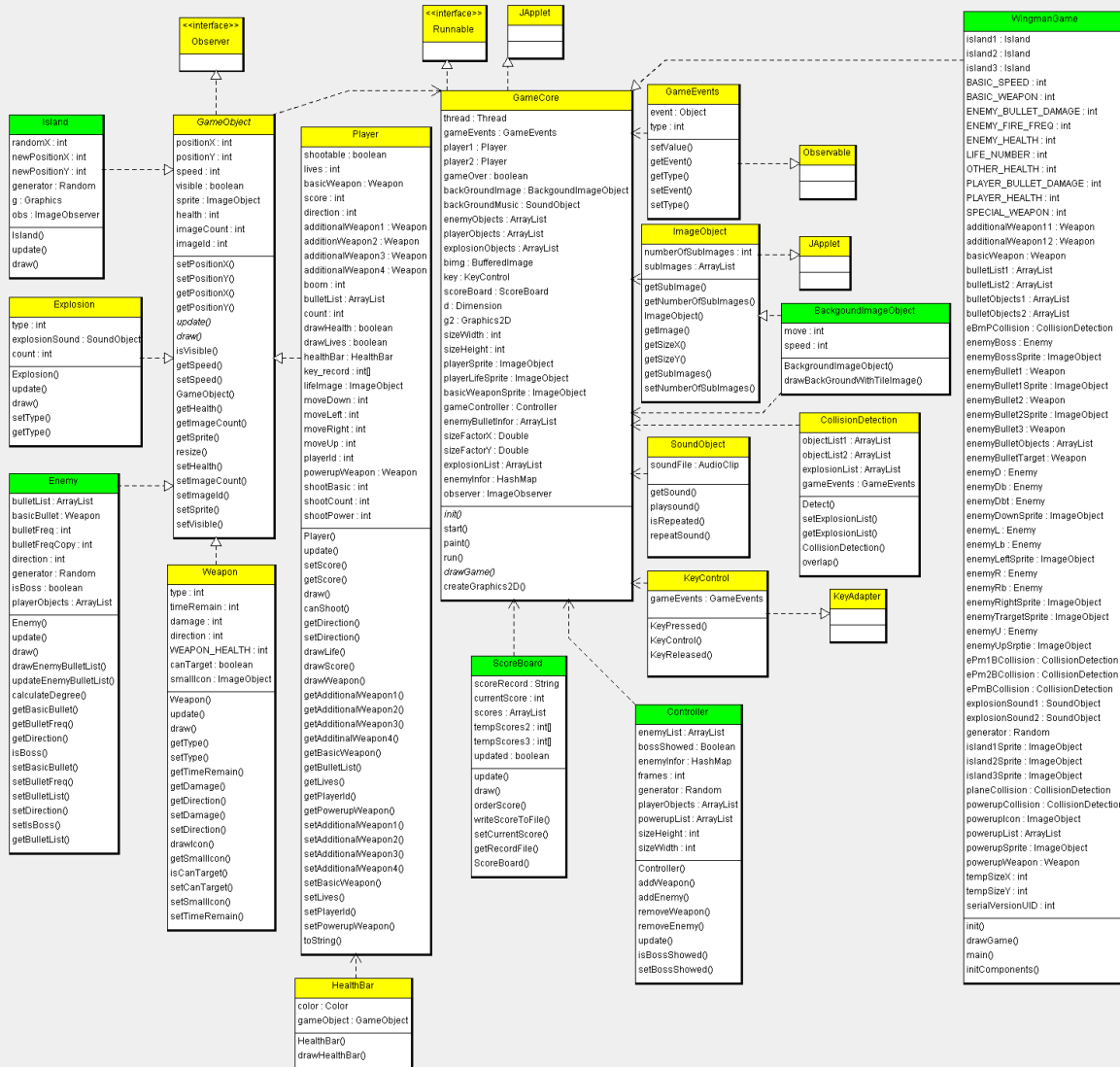
Tank

- Tank angle & moving direction handling
- Player viewing Window with mini-map
- Explosion effect and sound
- Soft walls (disappearing and appearing over time)
- Stronger Bullets when player picks up bonus
- Strong tank types when player picks up bonus
- Increase HP value when player picks up bonus
- Show strong weapon icon and remaining time to use
- Scores
- Health bar
- Map reading from file
- Background music
- 2 more spare life

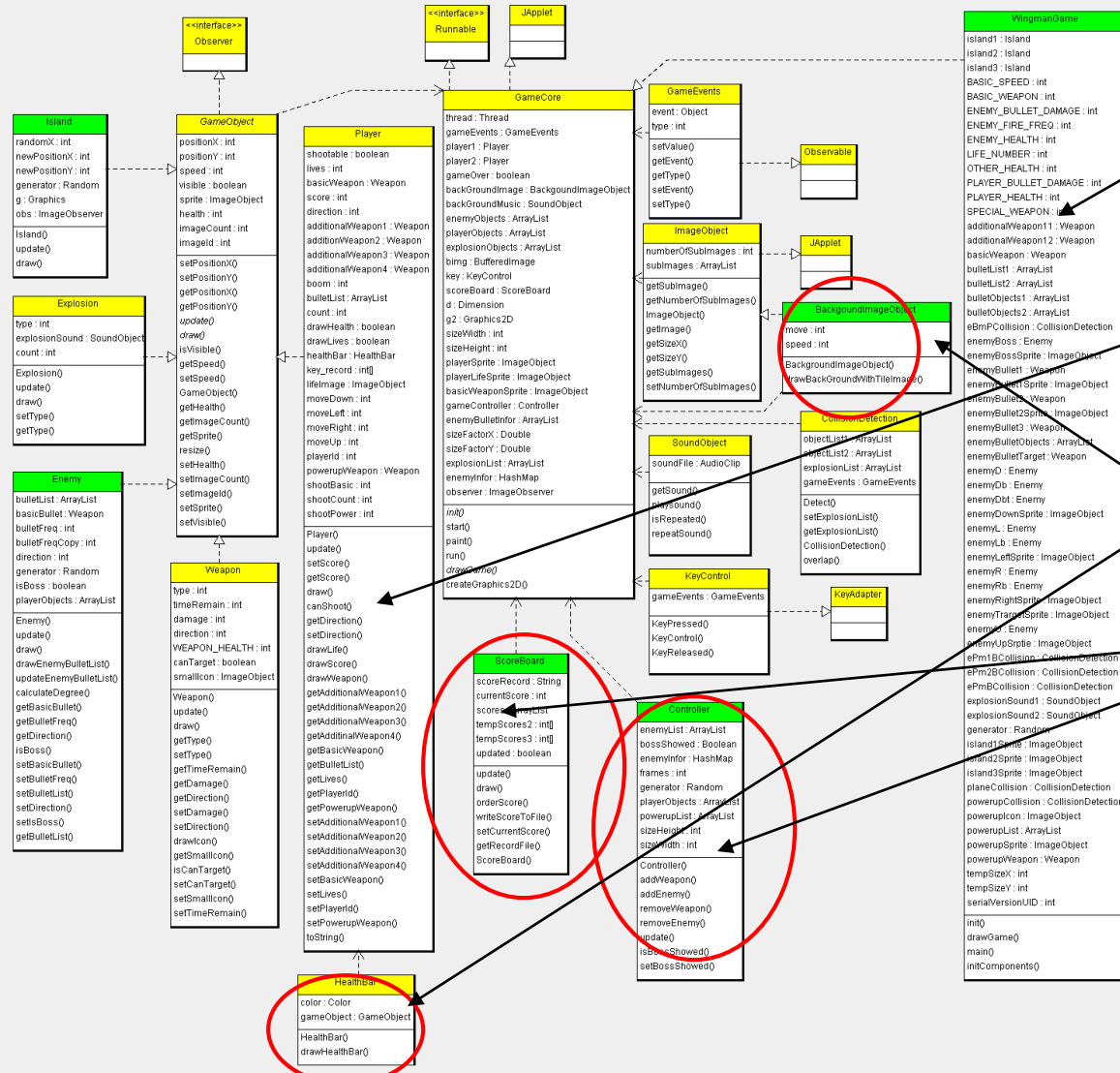
```

classDiagram
    class GameCore {
        <<interface>>
        Runnable
    }
    class GameCore {
        Thread thread
        GameEvents gameEvents
        Random generator
        Player player1
        Player player2
        boolean gameOver
        ImageObserver observer
        int life
        ImageObject backGroundImage
        Weapon weaponBasic
        ImageObject backGroundMusic
        ArrayList enemyObjects
        ArrayList playerObjects
        ArrayList enemyBulletObjects
        ArrayList playerBulletObjects
        ArrayList explosionObjects
        BufferedImage bimg
        KeyControl key
        long startTime
        File scoreFile
        ScoreBoard scoreBoard
        Sprite sprite
        long startTime
        File scoreFile
        ScoreBoard scoreBoard
    }
    class GameObject {
        int positionX
        int positionY
        int speed
        boolean visible
        ImageObject sprite
        setPositionX()
        setPositionY()
        getPositionX()
        getPositionY()
        update()
        draw()
        isVisible()
        getSpeed()
        setSpeed()
    }
    class Player {
        int health
        boolean shootable
        int lives
        Weapon basicWeapon
        Weapon secondaryWeapon
        int score
        int direction
        Player()
        update()
        setHealth()
        getHealth()
        setScore()
        getScore()
        draw()
        canShoot()
        drawHealthBar()
        drawWeapon()
        getDirection()
        setDirection()
    }
    class Weapon {
        int type
        int timeRemain
        int damage
        int direction
        Weapon()
        update()
        draw()
        getType()
        setType()
        getTimeRemain()
        getDamage()
        getDirection()
        setDamage()
        setDirection()
    }
    class Island {
        int randomX
        Island()
        update()
        draw()
    }
    class Explosion {
        int type
        SoundObject sound
        int count
        Explosion()
        update()
        draw()
        setType()
        getType()
    }
    class Enemy {
        int health
        Enemy()
        update()
        draw()
        getHealth()
        setHealth()
    }
    class ScoreBoard {
        File scoreRecord
        int currentScore
        update()
        draw()
        orderScore()
        writeScoreToFile()
        setCurrentScore()
        getRecordFile()
    }
    class ImageObject {
        int sizeX
        int sizeY
        int numberOfSubImages
        ArrayList subImages
        drawBackgroundWithTileImage()
        getSubImage()
        getNumberOfSubImages()
    }
    class GameEvents {
        Object event
        int type
        GameEvents()
        setValue()
        getValue()
    }
    class SoundObject {
        AudioClip soundFile
        boolean repeatSound
        SoundObject()
        getSound()
        playSound()
        isRepeated()
    }
    class CollisionDetection {
        ArrayList objectList1
        ArrayList objectList2
        Detect()
    }
    class KeyControl {
        KeyAdapter keyAdapter
        KeyPressed()
        KeyBoardInput()
        KeyReleased()
    }
    class KeyAdapter {
    }
    class Observer {
        <<interface>>
    }
    class Observable {
    }
    class WingmanGame {
        Island island1
        Island island2
        Island island3
        GameController gameController
        init()
        drawGame()
        main()
        initComponents()
        drawScoreBoard()
    }
    class Controller {
        ArrayList weaponList
        ArrayList enemyList
        int randomNumber
        Controller()
        addWeapon()
        addEnemy()
        removeWeapon()
        removeEnemy()
        update()
    }
    GameCore <|-- WingmanGame
    GameCore <|-- Controller
    GameCore <|.. GameObject
    GameCore <|.. Player
    GameCore <|.. Weapon
    GameCore <|.. ImageObject
    GameCore <|.. GameEvents
    GameCore <|.. SoundObject
    GameCore <|.. CollisionDetection
    GameCore <|.. KeyControl
    GameCore <|.. KeyAdapter
    GameCore <|.. Observer
    GameCore <|.. Observable
    GameObject <|.. Island
    GameObject <|.. Explosion
    GameObject <|.. Enemy
    GameObject <|.. Weapon
    GameObject <|.. ScoreBoard
    GameObject <|.. ImageObject
    GameObject <|.. GameEvents
    GameObject <|.. SoundObject
    GameObject <|.. CollisionDetection
    GameObject <|.. KeyControl
    GameObject <|.. KeyAdapter
    GameObject <|.. Observer
    GameObject <|.. Observable
    
```

Wingman Design diagrams (2)



Wingman Design diagrams (2)



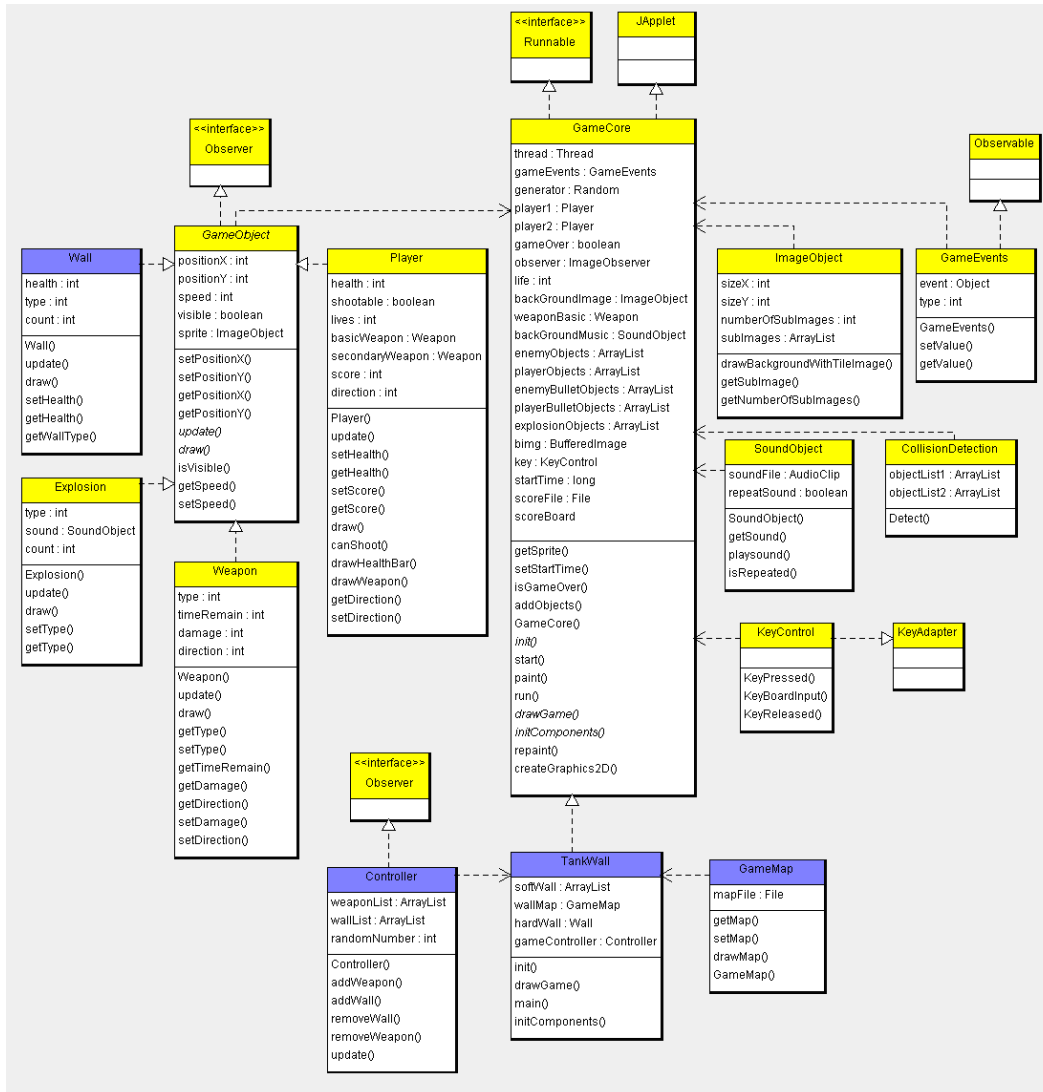
More variables

More methods

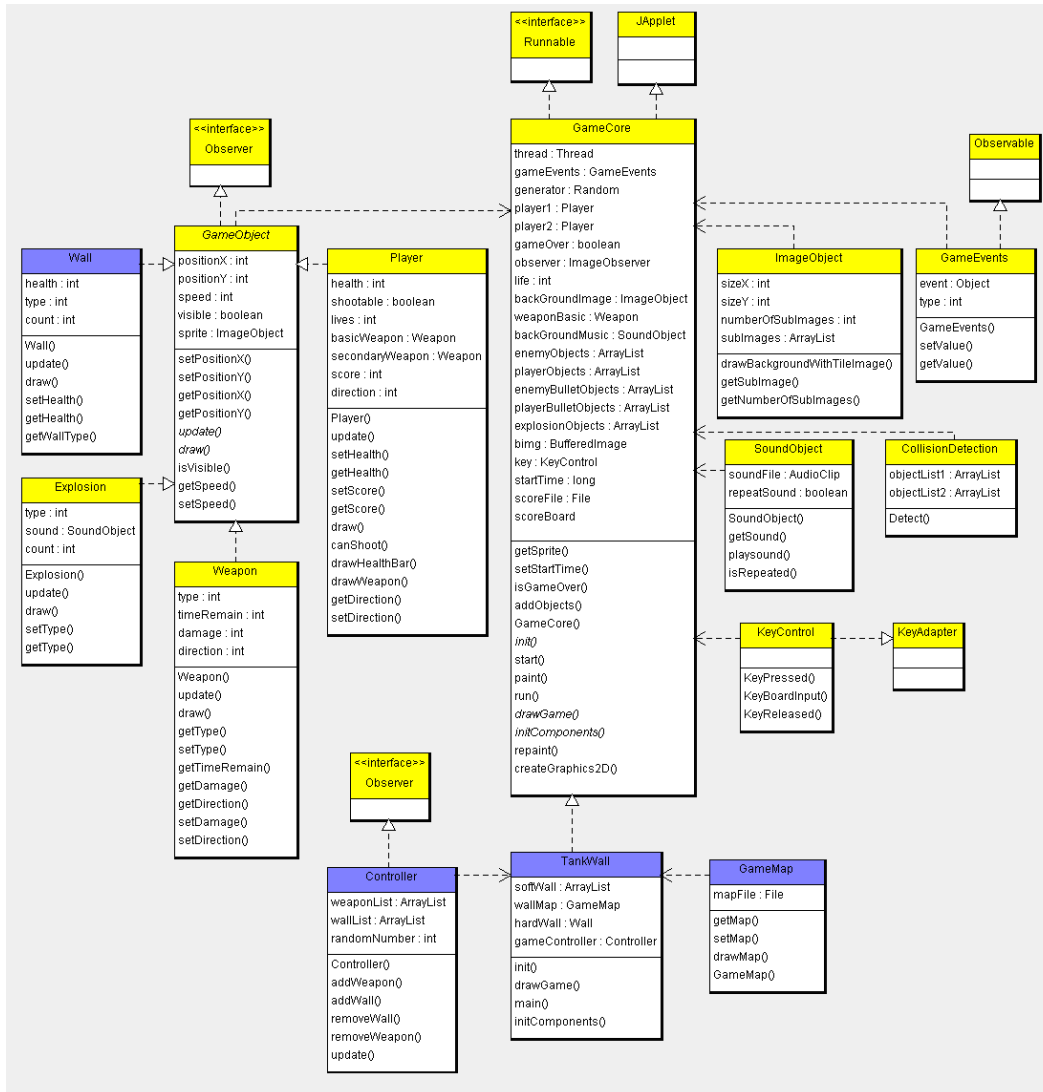
New class

Change of inheritance

Tank Design diagrams (1)



Tank Design diagrams (2)



More variables

More methods

New class

-HealthBar

-TankPlayer

-BackgroundImageObject

-SourceReader

Change of inheritance

-Controller

Reusability

Class reused from Wingman

- CollisionDetection
- GameCore
- GameEvents
- HealthBar
- KeyControl
- Explosion
- GameObject
- Player
- Weapon
- BackgroundImageObject
- ImageObject
- SoundObject

Singleton

- Subimages of each weapon are the same
- The one to choose is different
- Consider singleton (static)

From JAVA application to JAVA applet

Some things to consider for applet

- File read and write
- Path
- Permission issues
- Sign applet

Things to improve

- Collision Detection
- Uniform way to handle subimages
- Different weapons (new subclass or other..)
- Subroutines to new class, increase reusability
- Consider more about extensibility at the stage of design
- More coding, more experience help the design