Group 15
Assignment 2
Bubble Bobble

20-Time

For our own improvement, we decided to look at our code quality. For this purpose, a few main improvement targets were identified:

- Testing coverage improvement: as a guideline, a line coverage of around 75% (using Cobertura) was used for this. During this sprint, our test coverage went from a total of 33% to 36%. However the meaningful test coverage is a lot higher. As per example: our coverage for our objects package is 57%, but the meaningful test coverage will be close to 100%. So practically our meaningful test coverage will be a fair bit higher than 36%
- Removing bugs: bugs that were identified in the previous sprint were removed:
    - Objects fell off screen when window focus was lost;
    - Bubbles never disappeared, even when off screen, posing a memory leak;
    - One bubble was able to catch two enemies;
    - The player kept moving while dead.
- Checkstyle: using maven reports, the general readability of the code was improved by reducing the total Checkstyle errors from 558 to 360. However a lot of the errors that are left are due to not having Java Docs for variables and ,according to Checkstyle, not having the right scope on a lot of our methods and variables. These warnings will be disabled in the future, giving us a more realistic error count. The Java Docs for methods were also updated and expanded.

Furthermore, an analysis of our class and testing structure was done. A look at the UML diagram as generated by IntelliJ (**Fig 1)** provided a number of possible improvements to our code.

- All subclass instances have a logger field, yet none is defined in the superclass. This suggest an error in the current hierarchy, as all GameObjects should log their interaction/events. The logger should therefore become a property of the GameObject (abstract) class.
- Both the BubbleObject and the GravityObject have a SpeedX and SpeedY variable (though it being with a different name). This suggest that these classes are more Analogous than currently described in the code. Analyzing the objects as a whole, the conclusion can be made that the main property setting them apart is their reaction to gravity (or their difference in SpeedY per frame). In the case of immutable objects, their speed is simply always 0. A small adjustment in the immutable object's update method allows us to properly define the Speed variables in the GameObject class without disturbing the game objects behaviour.
- Finally, the diagram shows the tests could be improved by more testing methods and classes.

Implementing all those improvements and rendering the new UML diagram gives us **(Fig. 2)**, which follows SOLID principles better.

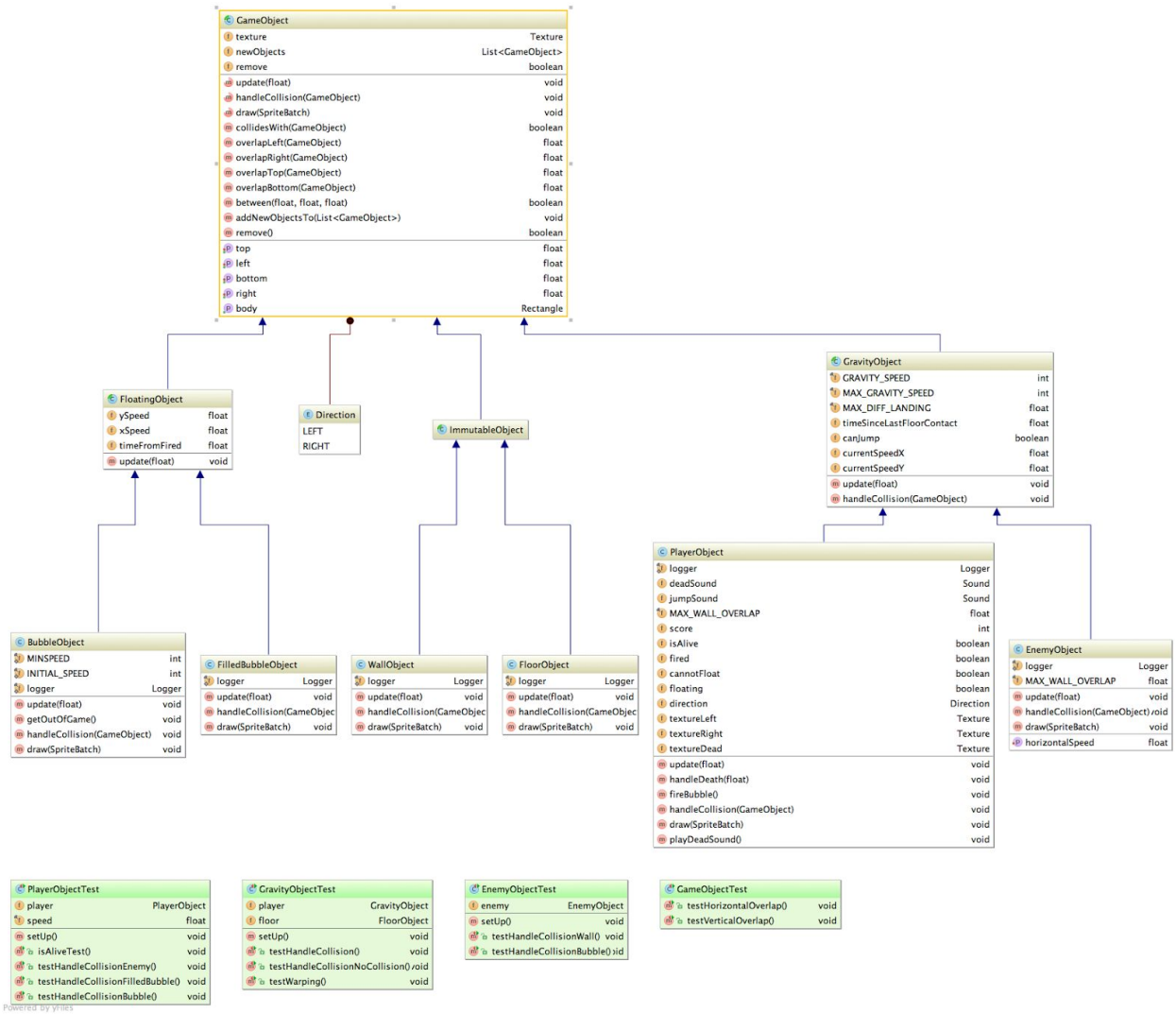
GameObject
texture Texture
newObjects List<GameObject>
remove boolean
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
collidesWith(GameObject) boolean
overlapLeft(GameObject) float
overlapRight(GameObject) float
overlapTop(GameObject) float
overlapBottom(GameObject) float
between(float, float, float) boolean
addNewObjectsTo(List<GameObject>) void
remove() boolean
top float
left float
bottom float
right float
body Rectangle
FloatingObject
ySpeed float
xSpeed float
timeFromFired float
update(float) void
Direction
LEFT
RIGHT
ImmutableObject
GravityObject
GRAVITY_SPEED int
MAX_GRAVITY_SPEED int
MAX_DIFF_LANDING float
timeSinceLastFloorContact float
canJump boolean
currentSpeedX float
currentSpeedY float
update(float) void
handleCollision(GameObject) void
BubbleObject
MINSPEED int
INITIAL_SPEED int
logger Logger
update(float) void
getOutOfGame() void
handleCollision(GameObject) void
draw(SpriteBatch) void
FilledBubbleObject
logger Logger
update(float) void
handleCollision(GameObjec
draw(SpriteBatch) void
WallObject
logger Logger
update(float) void
handleCollision(GameObjec
draw(SpriteBatch) void
FloorObject
logger Logger
update(float) void
handleCollision(GameObjec
draw(SpriteBatch) void
PlayerObject
logger Logger
deadSound Sound
jumpSound Sound
MAX_WALL_OVERLAP float
score int
isAlive boolean
fired boolean
cannotFloat boolean
floating boolean
direction Direction
textureLeft Texture
textureRight Texture
textureDead Texture
update(float) void
handleDeath(float) void
fireBubble() void
handleCollision(GameObject) void
draw(SpriteBatch) void
playDeadSound() void
EnemyObject
logger Logger
MAX_WALL_OVERLAP float
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
horizontalSpeed float
PlayerObjectTest
player PlayerObject
speed float
setUp() void
isAliveTest() void
testHandleCollisionEnemy() void
testHandleCollisionFilledBubble() void
testHandleCollisionBubble() void
GravityObjectTest
player GravityObject
floor FloorObject
setUp() void
testHandleCollision() void
testHandleCollisionNoCollision() void
testWarping() void
EnemyObjectTest
enemy EnemyObject
setUp() void
testHandleCollisionWall() void
testHandleCollisionBubble() void
GameObjectTest
testHorizontalOverlap() void
testVerticalOverlap() void

Fig 1.

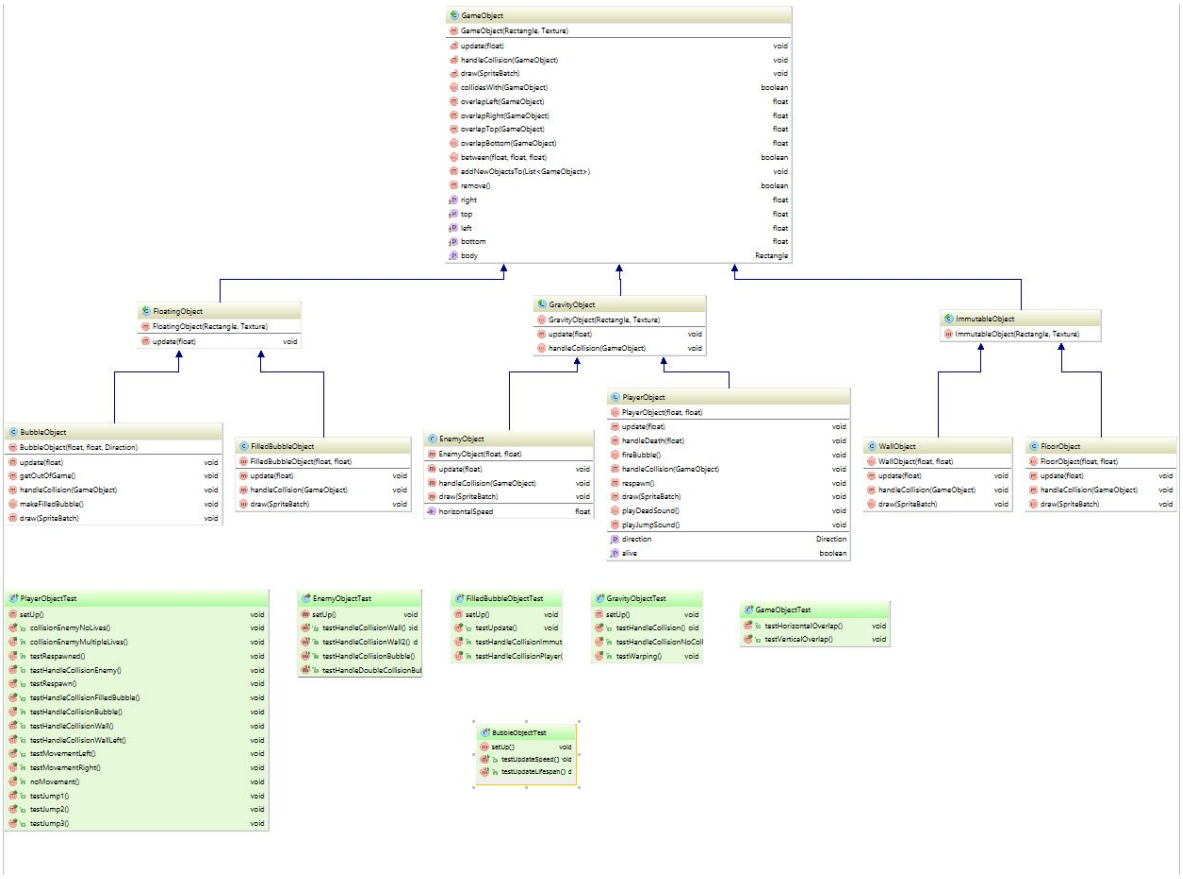
GameObject
GameObject(Rectangle, Texture)
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
collidesWith(GameObject) boolean
overlapLeft(GameObject) float
overlapRight(GameObject) float
overlapTop(GameObject) float
overlapBottom(GameObject) float
between(float, float, float) boolean
addNewObjectsTo(List<GameObject>) void
remove() boolean
right float
top float
left float
bottom float
body Rectangle
FloatingObject
FloatingObject(Rectangle, Texture)
update(float) void
GravityObject
GravityObject(Rectangle, Texture)
update(float) void
handleCollision(GameObject) void
ImmutableObject
ImmutableObject(Rectangle, Texture)
BubbleObject
BubbleObject(float, float, Direction)
update(float) void
getOutOfGame() void
handleCollision(GameObject) void
makeFilledBubble() void
draw(SpriteBatch) void
FilledBubbleObject
FilledBubbleObject(float, float)
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
EnemyObject
EnemyObject(float, float)
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
horizontalSpeed float
PlayerObject
PlayerObject(float, float)
update(float) void
handleDeath(float) void
fireBubble() void
handleCollision(GameObject) void
respawn() void
draw(SpriteBatch) void
playDeadSound() void
playJumpSound() void
direction Direction
alive boolean
WallObject
WallObject(float, float)
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
FloorObject
FloorObject(float, float)
update(float) void
handleCollision(GameObject) void
draw(SpriteBatch) void
PlayerObjectTest
setUp() void
collisionEnemyNoLives() void
collisionEnemyMultipleLives() void
testRespawned() void
testHandleCollisionEnemy() void
testRespawn() void
testHandleCollisionFilledBubble() void
testHandleCollisionBubble() void
testHandleCollisionWall() void
testHandleCollisionWallLeft() void
testMovementLeft() void
testMovementRight() void
noMovement() void
testJump1() void
testJump2() void
testJump3() void
EnemyObjectTest
setUp() void
testHandleCollisionWall() void
testHandleCollisionWall2() void
testHandleCollisionBubble() void
testHandleDoubleCollisionBu
FilledBubbleObjectTest
setUp() void
testUpdate() void
testHandleCollisionImmut
testHandleCollisionPlayer(
GravityObjectTest
setUp() void
testHandleCollision() void
testHandleCollisionNoColl
testWarping() void
GameObjectTest
testHorizontalOverlap() void
testVerticalOverlap() void
BubbleObjectTest
setUp() void
testUpdateSpeed() void
testUpdateLifespan() void

Fig 2.

Your wish is my command

For this part of the assignment, three new features were asked to be implemented:

- The player should be able to jump on bubbles;
- Warping should be implemented;
- Multiple levels should be implemented.

For the jumping on bubbles, the helper method overlapTop as defined in the GameObject class is used, which provides us with a way to check if two game objects are overlapping. The player has to have its canJump property set to true in order to jump, and should only be able to jump if coming from the top of the bubble. Therefore, adding the following lines in the handle collision method of the PlayerObject:

```
if (other instanceof BubbleObject) {
   if (between(other.overlapTop(this), 0, 5)) {
      canJump = true;
      logger.log("Player touched bubble");
   }
}
```

Relatively simply implements this feature.

For warping it's important to note that all gravity objects should be able to warp (namely enemies and the player). Therefore, in the GravityObject class a check should be made if the object has fallen below the screen, and if this the case, they should be replaced above the screen in order to fall into the screen again. For this, the following lines in the update method of the GravityObject implement this feature:

```
if (getTop() <= 0) {
   setBottom(Gdx.graphics.getHeight());
}
```

once again using the helper methods as defined in the GameObject class.

For multiple levels, a multiple of classes was implemented (Fig.3): first, a way to store save and read the levels was needed. This was divided in two classes:

- The LevelParser, which has the responsibility to read text files and form a list of GameObjects from them;
- Level, which contains a GameObject list and helper methods, such as levelFinished();
- A Leveleditor, which is a seperate main() that allows us to visually build levels.
- The LogicController was adjusted to contain a Level levelMap, to be able to switch levels on the fly.


Logger
loggers HashMap<String, Logger>
writer BufferedWriter
clazz String
getLogger(String) Logger
log(String) void
Package objects
«create»
LevelEditor
batch SpriteBatch
bitmapFont BitmapFont
objects List<GameObject>
currentlySelected Selected
create() void
resize(int, int) void
render() void
checkKeyboardInput() void
save() void
writeLineToFile(BufferedWriter, GameObject) void
renderObjects() void
checkMouseInput() void
checkUIClick(int, int) void
addSprites() void
pause() void
resume() void
dispose() void
checkCollision(GameObject) GameObject
LevelParser
parse(FileHandle) List<GameObject>
getObject(String) GameObject
Level
levelFinished() boolean
map List<GameObject>
Selected
WALL
FLOOR
EMPTY
LogicController
gameObjects List<GameObject>
player PlayerObject
levelMap Level
currentLevel int
MAX_LEVEL int
init(int) void
addGameObject(GameObject) void
setPlayer() void
readMap(int) void
loop(float, SpriteBatch) void
checkForLose() boolean
update(float) void
checkCollisions() void
handleNewObjects() void
removeObjects() void
draw(SpriteBatch) void
BubbleBobble
SPRITE_SIZE int
controller LogicController
batch SpriteBatch
level Level
scoreString String
bitmapFont BitmapFont
logger Logger
create() void
resize(int, int) void
render() void
pause() void
resume() void
dispose() void
Powered by yFiles

Fig. 3