Assignment 2

Bubble Spinner Group 20
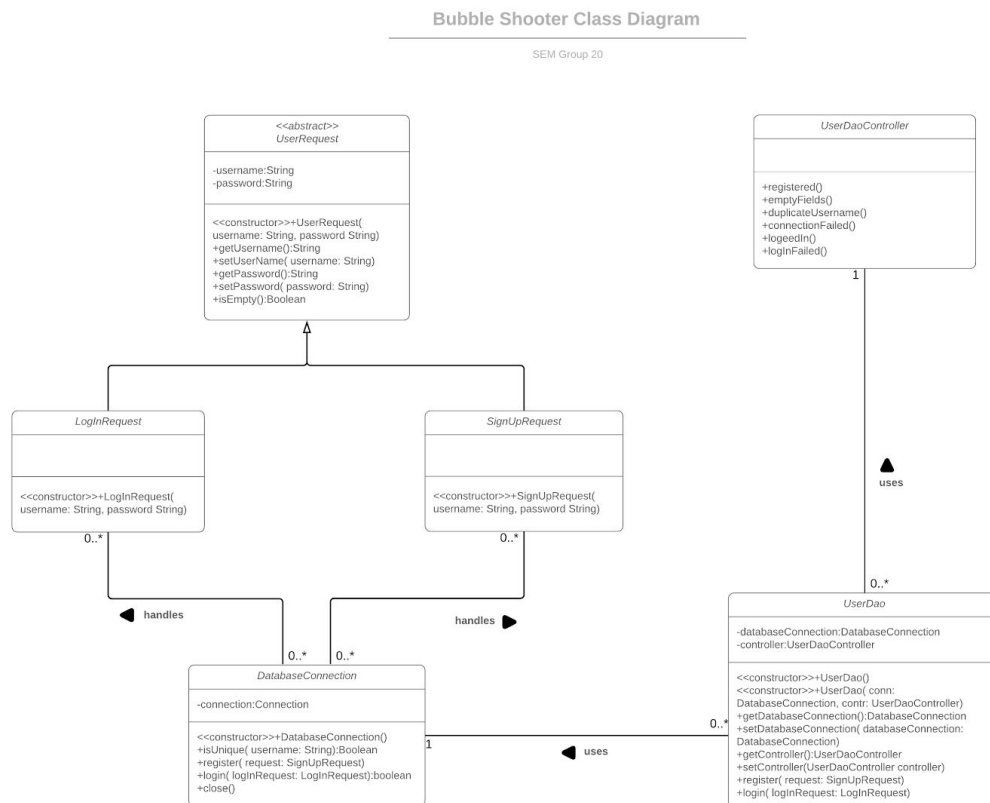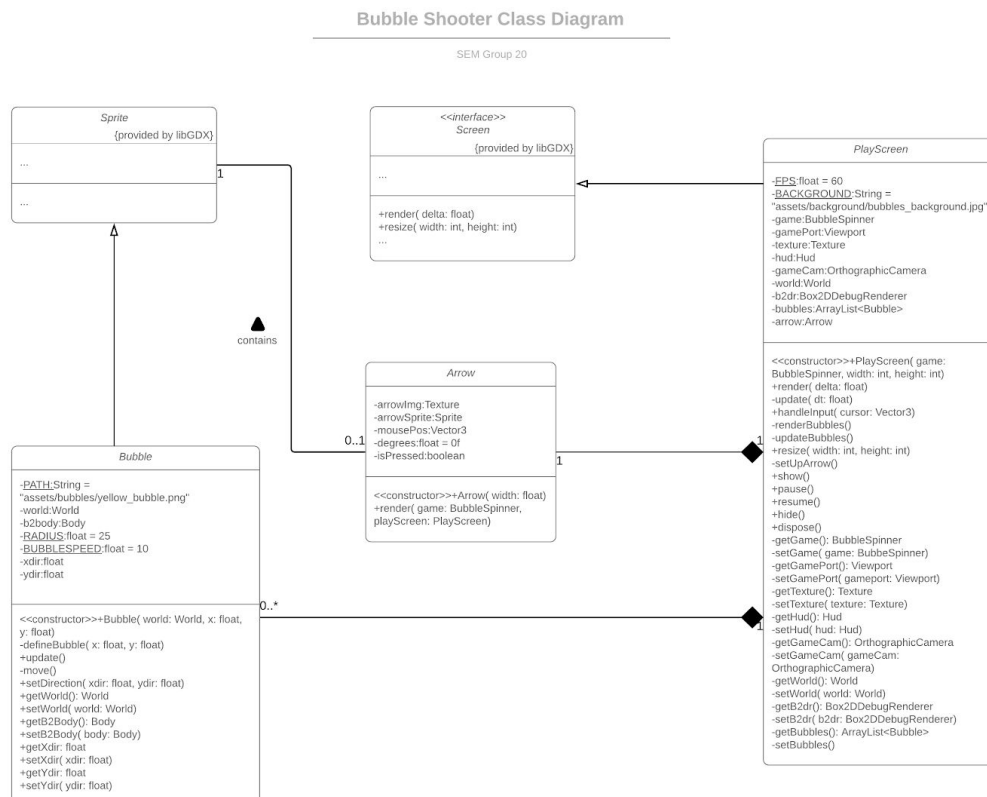
Contents

1. Class diagram

1.1 Authentication

The bubble shooter game requires authentication before a game can be started. The GUI screens for this are omitted in the diagram below. Credentials entered by the user are stored in a UserRequest. The UserDao class registers users by passing SignUpRequests to its DatabaseConnection, which stores new credentials into the database. The UserDao communicates back to the user through its UserDaoController instance, which displays pop-up messages. The UserDao can also pass LoginRequests to its DatabaseConnection, which then validates the credentials. Again, feedback is returned to the user by means of a UserDaoController instance. Both SignUpRequest and LoginRequest extend the abstract class UserRequest, since they have the same fields and methods but are appropriately used for different purposes. The diagram below is also available as a pdf in the same folder as this assignment.

Bubble Shooter Class Diagram

SEM Group 20

<>
UserRequest

-username:String
-password:String

<<constructor>>+UserRequest(
username: String, password String)
+getUsername():String
+setUserName(username: String)
+getPassword():String
+setPassword(password: String)
+isEmpty():Boolean

UserDaoController

+registered()
+emptyFields()
+duplicateUsername()
+connectionFailed()
+logeedIn()
+logInFailed()

LogInRequest

<<constructor>>+LogInRequest(
username: String, password String)

SignUpRequest

<<constructor>>+SignUpRequest(
username: String, password String)

uses

DatabaseConnection

-connection:Connection

<<constructor>>+DatabaseConnection()
+isUnique(username: String):Boolean
+register(request: SignUpRequest)
+login(logInRequest: LogInRequest):boolean
+close()

UserDao

-databaseConnection:DatabaseConnection
-controller:UserDaoController

<<constructor>>+UserDao()
<<constructor>>+UserDao(conn:
DatabaseConnection, contr: UserDaoController)
+getDatabaseConnection():DatabaseConnection
+setDatabaseConnection(databaseConnection:
DatabaseConnection)
+getController():UserDaoController
+setController(UserDaoController controller)
+register(request: SignUpRequest)
+login(logInRequest: LogInRequest)
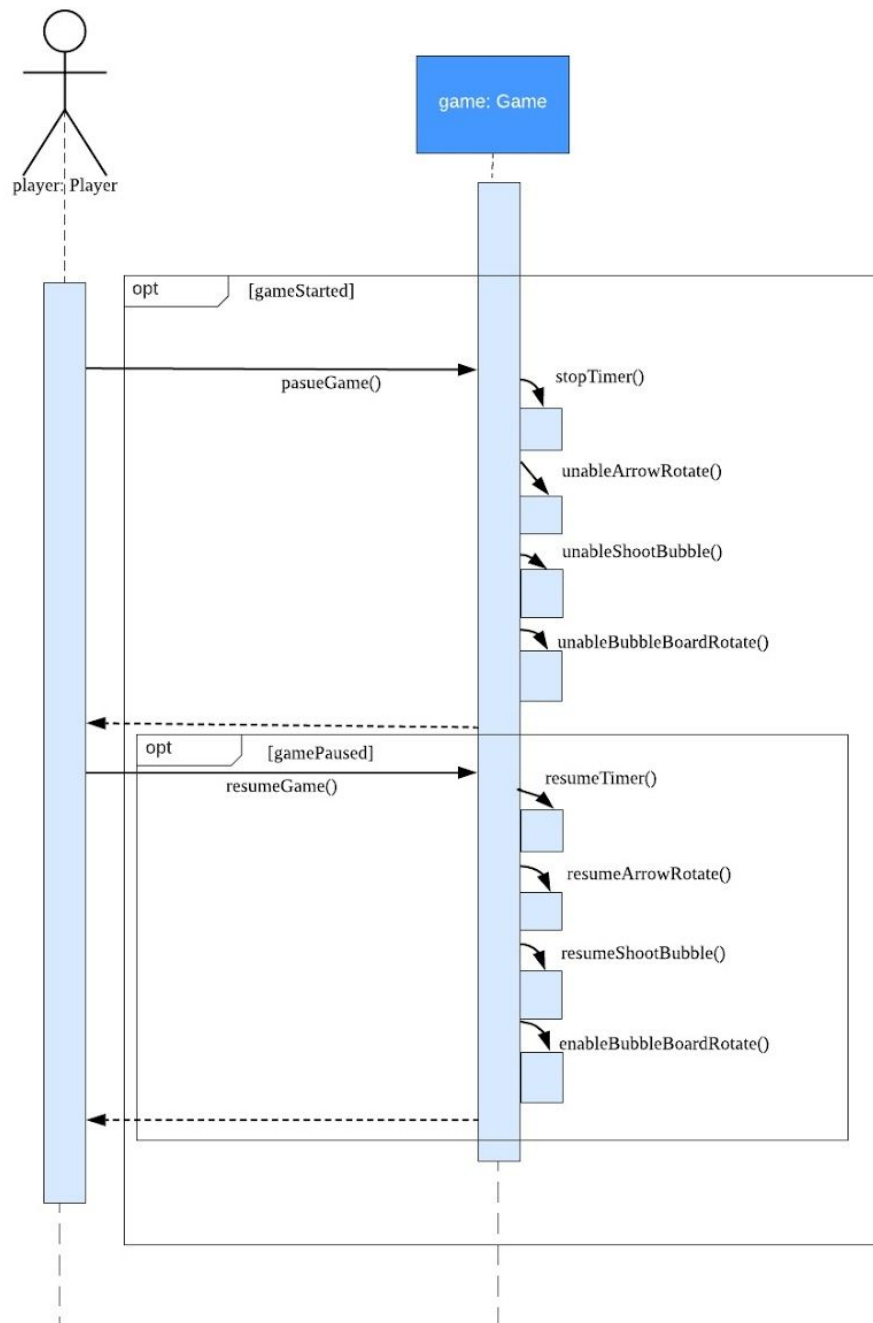
handles

handles

uses

1.2 Game flow

At this point, the game does not include much logic yet, since the functionalities are limited. Therefore GUI parts are not fully committed in this class diagram. Even some classes provided by the libGDX library are included to give an extensive overview of the prototype. The game objects Arrow and Bubble are part of PlayScreen. When the user switches to another screen, the bubbles and arrow will disappear as well. The Arrow class contains a Sprite instance as an attribute, while the Bubble class extends Sprite.
The Arrow class contains methods to follow the movement of the curso. The Bubble class contains methods to be moved. The PlayScreen can render and update its array of bubbles. The diagram below is also available as a pdf in the same folder as this assignment.
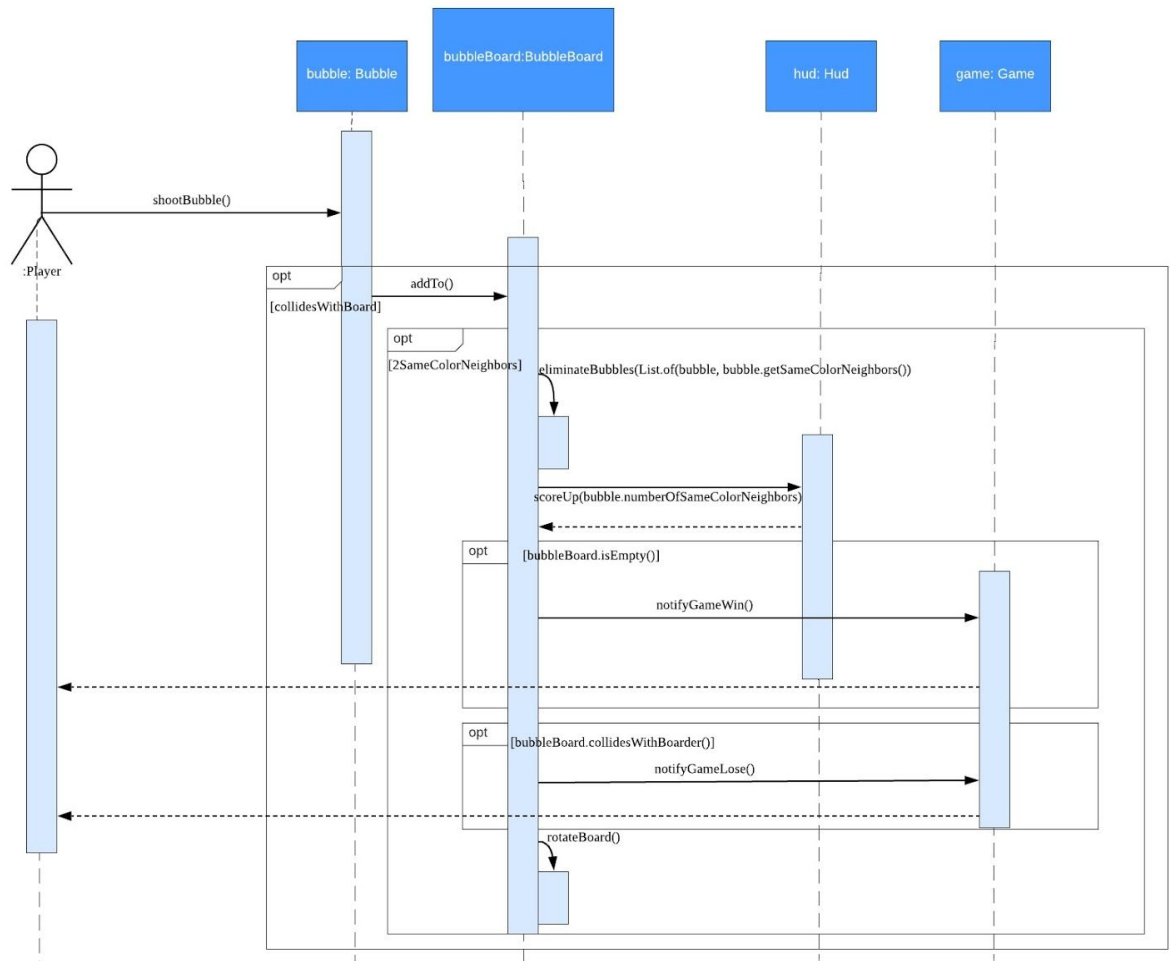
Bubble Shooter Class Diagram

SEM Group 20

Sprite
{provided by libGDX}
...
...

**<<interface>>
Screen**
{provided by libGDX}
...
+render(delta: float)
+resize(width: int, height: int)
...

PlayScreen
-FPS:float = 60
-BACKGROUND:String = "assets/background/bubbles_background.jpg"
-game:BubbleSpinner
-gamePort:Viewport
-texture:Texture
-hud:Hud
-gameCam:OrthographicCamera
-world:World
-b2dr:Box2DDebugRenderer
-bubbles:ArrayList<Bubble>
-arrow:Arrow

<<constructor>>+PlayScreen(game: BubbleSpinner, width: int, height: int)
+render(delta: float)
-update(dt: float)
+handleInput(cursor: Vector3)
-renderBubbles()
-updateBubbles()
+resize(width: int, height: int)
-setUpArrow()
+show()
+pause()
+resume()
+hide()
+dispose()
-getGame(): BubbleSpinner
-setGame(game: BubbleSpinner)
-getGamePort(): Viewport
-setGamePort(gameport: Viewport)
-getTexture(): Texture
-setTexture(texture: Texture)
-getHud(): Hud
-setHud(hud: Hud)
-getGameCam(): OrthographicCamera
-setGameCam(gameCam: OrthographicCamera)
-getWorld(): World
-setWorld(world: World)
-getB2dr(): Box2DDebugRenderer
-setB2dr(b2dr: Box2DDebugRenderer)
-getBubbles(): ArrayList<Bubble>
-setBubbles()

contains

Arrow
-arrowImg:Texture
-arrowSprite:Sprite
-mousePos:Vector3
-degrees:float = 0f
-isPressed:boolean

<<constructor>>+Arrow(width: float)
+render(game: BubbleSpinner, playScreen: PlayScreen)

Bubble
-PATH:String = "assets/bubbles/yellow_bubble.png"
-world:World
-b2body:Body
-RADIUS:float = 25
-BUBBLESPEED:float = 10
-xdir:float
-ydir:float

<<constructor>>+Bubble(world: World, x: float, y: float)
-defineBubble(x: float, y: float)
+update()
-move()
+setDirection(xdir: float, ydir: float)
+getWorld(): World
+setWorld(world: World)
+getB2Body(): Body
+setB2Body(body: Body)
+getXdir: float
+setXdir(xdir: float)
+getYdir: float
+setYdir(ydir: float)

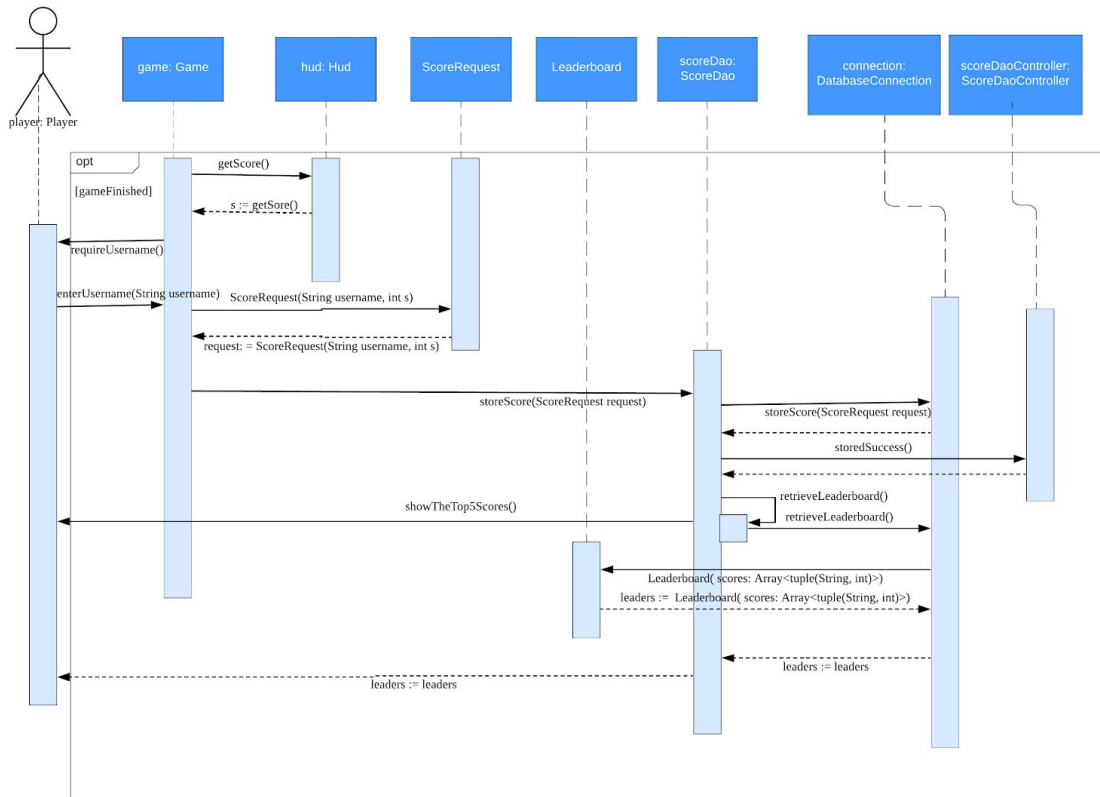1

0..1

1

1

1

0..*

2. Sequence diagrams

2.1 Pause and resume a game

2.2 Shoot a bubble

2.3

3. Working prototype

The first intermediate prototype of our Bubble Shooter game has the following features:

- **Authentication:** the system offers the options to sign up or sign in, while making use of a real database connection.
- A **new game** can be started: after authenticating, the player is able to initialize a new game. The timer will start counting and the score is set to zero.
- **Rotation the arrow**: the arrow at the bottom of the screen shall follow the movements of the cursor. This way, the player knows in which direction a ball will be shot.
- **Shooting a bubble**: the player is able to shoot a bubble by clicking their mouse. The bubble is shot in the direction of the arrow.