Assignment 2 group 27

Requirements for sprint #2
We should make requirements for the features we want to implement for the next sprint.
Each team member should first make requirements for his task and write them below here.

Different resolutions
Must have:
- The game must have no fixed resolution.
- The game must be playable on different resolution settings.

Should have:
- The game should look and function the same on different resolution settings

Could have:
- An option where the player could change the resolution used.

Restyle the GUI class (Sytze)
Must have:
- The new gui package must contain multiple classes.
- The new gui code must make it easier to switch to a different pane.
- The new gui code must be more readable.
- The classes in the gui package must have clear and split responsibilities.
- The new gui package should make use of oop-hierarchies.

Should have:
- The new gui should not have less functionality than the old gui.
- The new gui package should have classes for different panes.
- The new gui package should contain a MainGui class that should be responsible for switching panes and running the game.

Change the fish that the player controls to a shark (Stefan)
Must have:
- The new image must look like a shark

Should have:
- The new image should scale to a decent size
- The new image should make it clear that this is the player

Tests for the GUI class (Stefan)
Must have:
- The tests must cover at least 75% of the branches
- The tests must cover all gui classes

Should have:
- The tests should cover at least 50% of the methods that can be called.

Sound effects (Sytze)

Must have:
- The user must hear a sound when the shark eats a fish.
- The sound must be shorter than 2 seconds.

Restart the game (Michiel)

Must have:
- The user must be able to restart the game when he has been eaten by a fish.
- The user must be able to restart the game when he has won the game.
- The current score of the user must be set to zero after restarting the game..
- The size of the lasershark must be at his beginning size when the game is restarted.

Should haves:
- The lasershark should be in a sea without other fishes when the game is restarted.

Different fishes appear(Youri)

Must have:
- The fishes must not all use the same sprite.

Should have:
- The fishes should have at least 10 different sprites to choose from.

Make more tests for the highscore class(Daan)

Must have:
- Every method from the highscores shall be tested
- The overall coverage of the highscores class shall be higher than 95%

Restyle the highscore class(Daan)

Must have:
- The highscore class shall be more readable
- The highscore class shall have a singleton structure

New UML:

Options
+Options(Dimension)
+getInstance()
+destroyInstance()

MainGui
+start(Stage)
+browseTo(StandardPane)
+browseToGlobal(StandardPane)

LosingPane
+LosingPane()

StandardPane
+StandardPane()
+addBackGround()
+playMusic(String)
+playSoundEffect(String)
+showScore(String)
+addText(String, int, Position)
+addMidText(String, int, double)

GamePane
+GamePane()
+startGame()
+stopGame()
+resumeGame()
+showScore()
+clearPaneOfImageView()
+showShark(LaserShark)
+showFishList(List)
+fishImage(Swimmer)

WinPane
+WinPane()

HighScores
+readHighscore()
+writeHighscore()
+fixHighscoreCount(ArrayList)
+makeHighscoreString()
+increaseScore(Swimmer)

ScreenController
-GamePane currentPane
-FishController fishCon
+ScreenController(GamePane)
+getNextFrameInfo(double)
+start()
+operation()

DirectionInputController
-DirectionCallback callback
-Scene scene
+KeyboardController(ScreenController, DirectionCallback)
+keyPressed(KeyEvent)
+keyReleased(KeyEvent)
+handleInput()
+handle(KeyEvent)

Logger
+Logger(FileWriter)
+write(String, parameter)

RestartGameController
+RestartGameController()
+restartGame()
+handle()

<Enum> Direction
-North
-NorthEast
-East
-SouthEast
-South
-SouthWest
-West
-NorthWest
-None

FishController
-fishList
-LaserShark shark
-float fishSpawnChance
-Random rng
+FishController()
+addFish(Swimmer)
+updatePositions(double)
+clearFish()
+checkForCollisions()

<interface> Swimmer

Fish
-Position position
-float size
-double speed
-Direction direction
-boolean alive
+Fish(Position, float, double, Direction)
+increaseSize(float)
+move(double)
+collision(Fish)
+isOnScreen()
+kill()
+isAlive()
+makeHitbox()

Position
-double posX
-double posY
+updatePosition(Direction, double, parameter)
+calculateDistance(Position)
+onScreen(double)
+clipPosition(double, parameter)
+middlePosition()
+upperCornerPosition()

LaserShark
+LaserShark(Position, float, double, Direction)
+eat(Swimmer)
+move(double)
+kill()
+putDirection(Direction)

FishBot
+FishBot(Position, float, double, Direction)
+generateFish()
+generateFish(Random)

Enemy
+Enemy()

Powered By Visual Paradigm Community Edition