Design Document

Version 2.0 - 2021/04/13

QIX

Abdullah Aftab - 500948702

Deandra Spike-Madden - 500946801

My Chi Duong - 500641750

Maryam Elbeshbishy - 500943034

Igor Gonçalves Penedos - 500898291

Table of Contents

1 Introduction……..3

1.1 - Project Overview ……...3

1.2 - Requirement Changes ……...3

2 Diagrams……..4

2.1 - Object Classes ……...4

2.2 - Class Diagram ……....……....6

3 Testing Plan ……7

3.1 - Introduction ……...7

3.2 - Test Report……9

1 Introduction

1.1 - Project Overview

The objective of this project is to develop a recreation of the 1981 puzzle video game, Qix with as much functionality as possible. Qix involves three main components: the marker, the Qix and the sparx. Using the arrow keys the user takes control of the marker. The objective is for the marker to take over as much of the board without hitting the Qix (which moves in random order) or the Sparx (Which moves along the edges of the collected land). We decided to use Pygame, as it was simple to learn and our team was familiar with the python programming language. We also used OOP to structure the code in a clean and organized manner.

1.2 - Requirement Changes

Some requirements from the original game were omitted and are as follows. Rather than a PUSH command, we allowed the user to enter the unclaimed field through just the arrow keys. We also omitted the feature in which a push is cancelled and the marker is brought back to its edge when the Sparx reaches an unfinished push. To make the game more difficult, and balance out the previous omission, we decided to increase the amount of Sparx as the levels progress as well as make the Qix move a lot faster. We found that this change made the game feel more intense in later levels and allowed for a better scale in difficulty while still allowing for fun. We also allowed players (and some Sparx) to go along edges that get claimed over. This helped balance the difficulty in later levels, however some Sparx will stay along the original border so that the player may not sit on the original border for too long. The Qix is also as big as the player, this is meant to also balance out later rounds (specifically once the player reaches level 5) as the Qix moves at a very fast pace.

2 Diagrams

2.1 - Object Classes

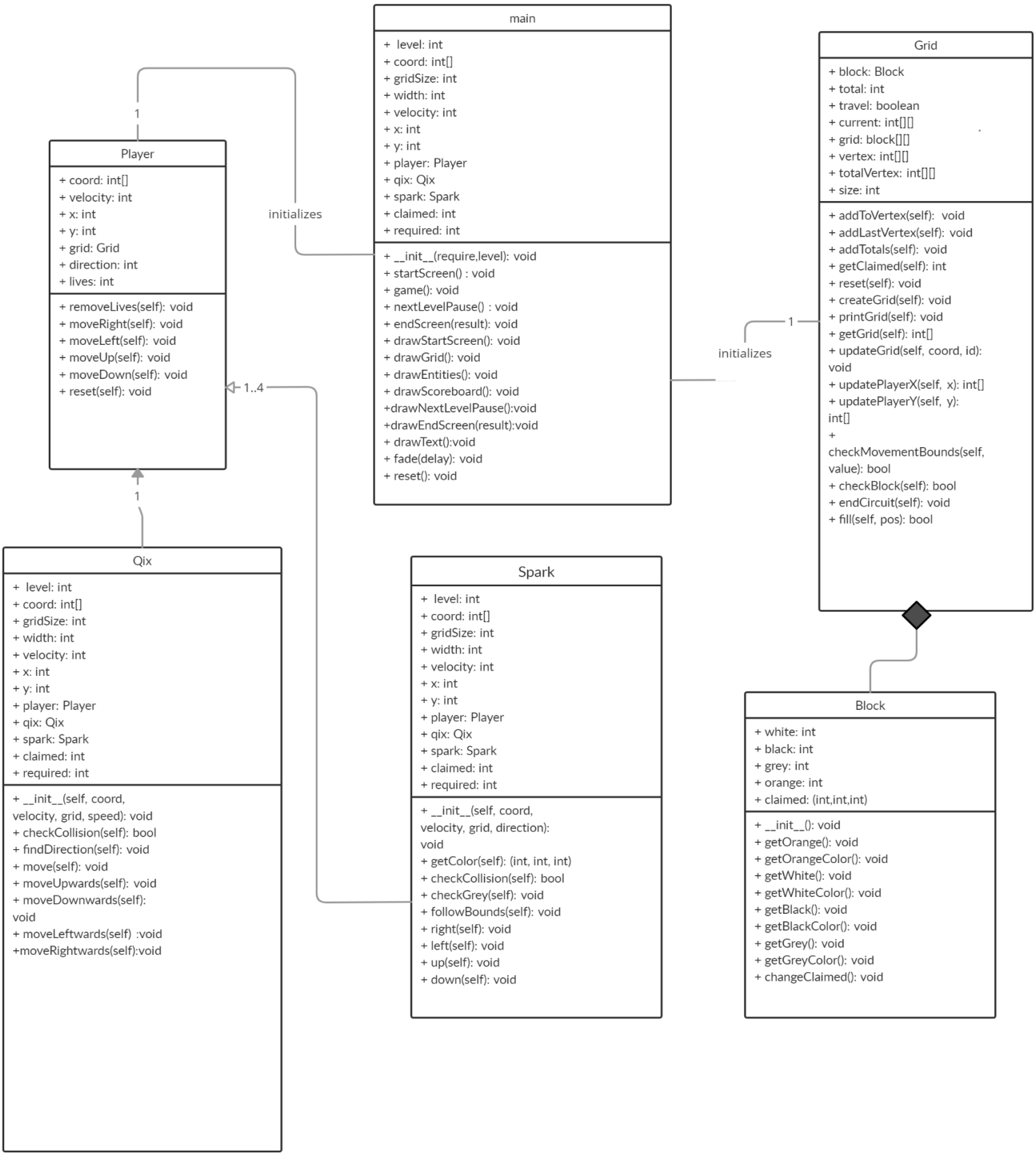
main	**Grid**	**Qix**
import pygame *import random* *import math* *from Block import Block* *from Grid import Grid* *from Player import Player* *from Spark import Spark* *from Qix import Qix* win gridSize velocity x, y block grid player level spark spark2 spark3 spark4	*from copy import deepcopy* block total travel coord current grid vertex totalVertex lastVertexCrossed totalMotion size createGrid()	*from Player import Player* speed coord velocity grid count direction collision
__init__(require, level) startScreen() game() nextLevelPause() endScreen() drawStartScreen()	__init__(size,block) addToVertex() addLastVertex() addTotals() getClaimed() reset()	__init__() checkCollision() findDirection() move() moveUpwards() moveDownwards()

drawGrid() drawEntities() drawScoreboard() drawNextLevelPause() drawEndScreen(result) drawText(font,font_Size, color, text_String,x,y) fade(delay) reset() main()	createGrid() printGrid() getGrid() updateGrid() updatePlayerX(x) updatePlayerY(x) checkMovementBounds(value) checkBlock() endCircuit() fill(pos)	moveLeftwards() moveRightwards()

Block	**Spark**	**Player**
import random white black grey orange claimed	*from Player import Player* coord velocity grid next direction isGrey	coord velocity x y grid direction lives
__init__() getOrange() getOrangeColor() getWhite() getBlack() getGrey() getWhiteColor() getBlackColor() getGreyColor() changeClaimed()	__init__(Player) checkCollision() checkGrey() followBounds() right() left() up() down()	__init__() removeLives() moveRight() moveLeft() moveUp() moveDown() reset()

2.2 - Class Diagram

**Note: Python does not private/public attributes and methods, thus all methods/attributes below are public*



3 Testing Plan

3.1 - Introduction

The Qix game will be tested for functionality by testing each of our classes through the development process to ensure the game was functioning to our liking. Our team tested both for general success and for meeting the requirements provided. Testing is done throughout the coding process and will be demonstrated through the video provided. When the result did not match the expected, the test will have been considered a failure and will be looked into for debugging in the next sprint. In this sprint, we focused on the basic functionality of our three state screens, the start, gameplay and end screen. The general functionality of each state screen will be tested to ensure that transitions between each state is as expected.

The start screen testing will be short. We will be focused on testing that the drawings we create are displayed in a proper manner and that the user is able to transition from the start screen to gameplay with any key inputted, as is stated in the instruction.

The gameplay is focused on the game itself and is the most important part of testing. The main methods and classes will be tested here, to make sure that gameplay is as expected. We will make sure that the player is able to move along the board, and our sparx moves along the edges of the claimed land. Testing for the Qix will also take place, ensuring that it moves in a randomized manner and collision with both Qix and Sparx is successful in removing a life and resetting player position. Land being claimed and filled will also be tested with various paths to ensure that the algorithm used works as we expect, and that no bugs occur when a user tries an obscure shape.

Gameplay also includes levels, and thus we will also be testing the functionality of each level, to ensure changes to the number of sparx or movement of the Qix does not affect gameplay in a negative way. Finally, we will be looking at the drawing of the scoreboard (on the right side of the screen) in which the percentage of land claimed and lives left are displayed. We will be testing that once a collision occurs a life is removed and that once land is filled the correct percentage is printed out to the user.

Testing the end screen will focus on making sure our “restart” functionality successfully transitions back to the start state of the game, and that you are able to play the game again. We will also be testing the drawing, similarly to the start screen, to ensure the level the user ends at and percentage claims are displayed in a clear and aesthetic manner. We will also be testing that the user is able to exit the game when the end screen is reached.

3.2 - Test Report

Functionality	Test Case	Description	Input	Expected	Result
Start Screen	drawStartScreen()	Test to determine that the title and instruction to start the game is written correctly on the screen	none	-	TRUE
	startScreen()	Test to determine that when any button is clicked the game will start.	Any key	Any key	TRUE
Gameplay	Spark/Qix collides with marker *(checkCollision())*	Test to ensure that the marker position is reset and a life is lost.	none	-	TRUE
	Qix Movement	Test to determine that when a user claims an area, the Qix moves in a random direction and stays outside of the claimed area.	none	-	TRUE
	Spark Movement	Test to determine that when user claims area, the Spark moves only on the outside of the claimed area and/or the border of the game	none	-	TRUE

	Marker movement	Test to ensure the user can use arrow keys to move the marker.	Arrow Keys	Arrow Keys	TRUE
	Area Claimed Fill	Test to ensure that when a user encloses an area, the enclosed area is filled with a color and the spark/Qix/marker cannot move through it.	none	-	TRUE
	drawScoreboard()	Tests to ensure that the area claimed percentage, user's lives, and current level is updated correctly	none	-	TRUE
	Levels	Tests to determine that game functionality stays the same as levels increases, the number of sparks increase accordingly, once all the required area is claimed the user is taken to an inbetween level screen that displays the area claimed and what level the player has completed.	none	-	TRUE
	User has zero lives left	Tests to determine that when user runs out of lives, the game is redirected to the end screen	none	-	TRUE
End Screen	Last Level	Test to ensure that the correct level is shown on the	none	-	TRUE

		screen when user completes/loses the game			
	Area Claimed Percentage	Test to ensure that the correct area claimed percentage is shown on the screen when user completes/loses the game	none	-	TRUE
	Restart	Test to determine that when any button is clicked on the end screen, the game redirects to the start screen and gains its functionality.	Arrow Keys	-	TRUE