Balance it!

Authors: Rohan Parikh, Akshat Adsule
Revision: 5/6/22

Introduction:
Our program is a game in which users attempt to balance various objects through their phones. Our game will be level-based with increasing difficulty and skill required. To increase the difficulty, we would have more storage spaces and more items to sort. Our program is aimed at people who have injuries or conditions in their hands that impede their fine motor skills. Through our game, users can practice their fine motor skills and regain their abilities. The rules are to categorize the various items into the same-colored bins. If an item is dropped or miscategorized, the user has lost and the game will reset. Once a user has successfully categorized all the items, they will win and move on to a more difficult level. Tilting the phone would change the surface angle which would move the blocks from side to side- trying to get it into the box. The final level will have the most storage boxes and most things to store and balance. The main target audience for our program is people with arthritis and this program will aid them in improving their hand-eye coordination along with other traits.

Instructions:

1) Find the runnable Jar file in the dist folder
    a) Alternatively you may build the jar yourself by running `./gradlew assemble`
2) Run the jar
    a) This should just work by double clicking the jar file, but you may need to run the `java -jar` command
    b) Alternatively, you can run `./gradlew run` to run the project
3) Profit
    a) Once the program launches, you should be greeted with a QR code. Scan the code with your phone, and you should see your phone's gyroscope data being reported on the screen. Keep track of the beta value, as this value should show up (and update) on the running java application.
        i) If you have an iPhone, you may need to use Chrome to open the URL

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER):
Must-have Features:

- Mobile-based controller. Controlling the game through a physical device is an integral part of our game. With our goal of helping those with limited mobility, a mobile-based controller is expressly crucial for such a goal. There will be an option to play specifically on the computer rather than having a phone controller.
- Networking. It would be much easier for the user if the process of playing the game was as simple as possible. Thus, having a network connection between the device and the computer

is important as it is the most seamless way of handling inter-device communication (compared to something like having a USB connection)
- Physics. Our game is based on balancing which is fundamentally physics-based. Being able to do this with accuracy and precision will make our game intuitive and grounded.
- Levels. We need to make several levels in order for the players to have options. Also, the same level over again could get repetitive and boring- even if this is designed to aid people with disabilities- so it would be a must to have level.
- Save feature. A save feature would be necessary if the players want to have a way to get back to the level they were at previously and not restart the whole game every time.

Want-to-have Features:
- Being able to go back to previous levels. If a user would like to practice a certain skill again in a level, they should be able to do so.
- Being able to skip. If a level is too difficult for someone, they should have the ability to skip the current level and move on to the next.
- Stars with time- 3 stars for under a certain time, lower stars if time was higher. Assigning star values will motivate the user to complete the level in a faster time and improve the fine motor skills.
- Handling networking errors effectively. Networking errors are bound to happen (such as client disconnects), and our game ideally should be able to handle such errors gracefully.
- Fancy UI. Ideally, our UI should be cohesive and easy to use.

Stretch Features:
- Multiplayer aspect with competition based levels/leaderboards. This would be a very nice feature to have (and is entirely possible with Firebase), it seems out of the timeframe we have for this project.
- 3D graphics and physics. While this is possible with Processing, we do not believe that we currently have the scope nor the time to fully develop 3D physics and handle that in processing.
- Level builder. Level building would introduce a whole new UI and flow we would have to design and implement, and is thus out of the time frame for this project.

Class List (basic for now, will expand once we start coding):
Main - Starts the program
Drawing surface- for the screen attributes and some methods including the draw method, mouse pressed.
LevelSuperclass - Contains data about the current level
Blocks class- Contains information about the actual objects that the user has to move to the balancing mechanism
Balancing aspect class- contains how velocity-gravity works with a seesaw or other mechanisms(changes with levels)

Networking - Handles establishing communication and receiving data from client devices.
Screens- Has all of the attributes for the screens including width, height, and the screen menu

Credits:

Developers

- Akshat: Networking, level design, and game code.
- Rohan: Physics, level design, and game code.

Libraries

- Processing
- Firebase
- QRGen
- Java Standard Library