Speedy Speed Bike Technical Document

Sam Fox | EDG220-02 | Team 1 | Sprint 2

Table of Contents

Delivery Platform

PC - Windows:

Speedy Speed Bike will be a PC exclusive for Windows operating systems. Doing this will be easier on the team during the development process as we will not have to design the game for multiple different platforms. The game is also intended to be played with only a Wii balance board and connecting the balance board to another game system, like a console, might be difficult especially for the player. Working with one platform also will reduce the amount of bugs during development as we do not have to configure the design for multiple operating systems. If we had more time to develop the game and the Wii was not as old as it was we could have also made the game playable on the Wii as well as PC. But because the Wii is so old and it might be hard to get a development kit for it we will not pursue that possibility. So overall a Windows release would be much faster and easier for our development team.

Development Environment

Git:

The version control system that we are using is Git. Git allows us to share the project files between all group members so we can work on the project easily. With Git each

team member can download (pull) the files from a remote repository and then upload (push) their work so other team members can use it. However, there are risks involved with Git. If a conflict occurs with some of the files it could be difficult to fix and may waste time and possibly set us back as a team, but if we work carefully nothing bad will happen. Using Redmine to set up our repository will allow the whole team to view all the files and what is changing, and for these reasons we are using Git.

GitKraken:

GitKraken is a GUI interface for working with Git. GitKraken makes it easier to visualize the repository and removes the need to use the Git command line. This makes Git less intimidating as it is made more for people new to Git who want to understand it more.

Unity:

For the game engine we will use Unity as it is a good platform for collaboration and easy to work with (version 2020.3.30). Unity is a professional level game engine that can do everything we need it to for this game. It is also widely used so finding help and documentation on it will be easier and will make solving problems faster. For these reasons Unity is the ideal engine for us to use to build this game.

Visual Studio 2019:

To do the C# coding required for this project we will be using Visual Studio 2019. One reason for this is because Unity works with Visual Studio and they pair well. Visual Studio also offers features for easy debugging with Unity projects. It is also very common and used broadly so finding online resources for help will be easy and help make production of the game faster.

Photoshop:

Photoshop is a standard program that is used by many artists within the gaming industry. It will be used to create the art assets as it offers many different features to

paint, size, and distort drawings with ease. Photoshop being a raster based program, it is used to create textured drawings to help add effect to the overall art appeal.

Ableton Live 10:

To handle the music side of things, Ableton Live 10 will be used in addition to Logic. Ableton has a large amount of digital instruments, and plug-ins are very easy to install and use. The plug-in that will be used for this project is LABS by Spitfire Audio, due to it being free and incredibly high quality.

Logic:

Logic will also be used for Music composition. Logic offers a whole suite of professional quality synthesizers and digital instruments. Additionally, this is the standard program for many professors, so if an issue should arise surrounding music, it will be easy to find help on campus.

Redmine:

Redmine is a project management tool which allows our team to make and maintain a wiki with information and documentation for our game. It also allows us to create tasks and log hours to those tasks so we can keep everything organized. We also use Redmine because of its remote repository which is how we share work on the project with one another using Git. This also helps us stay organized because we can clearly see one another's work and what changes have been made over time.

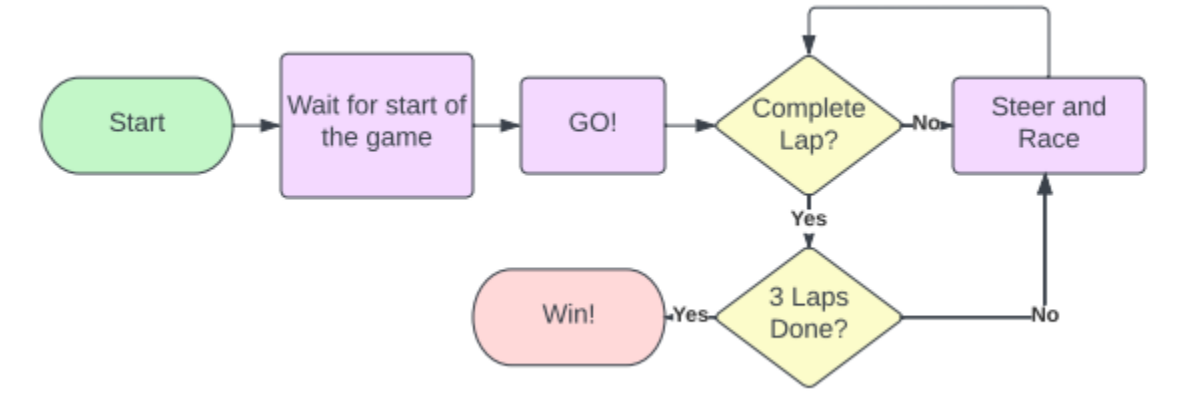
Mattermost:

Mattermost is a messaging service that we as a team use to communicate and reach one another about matters related to the project. When we need to reach one another regarding something for the project we use Mattermost to quickly reach one another. Also we are able to see when team members commit changes to our team repository. When we need to have virtual meetings we also use the built in Google Meet service to instantly join a meeting.

Google Meet:

While we plan to have all our meetings in person, we need a back up way to meet if we cannot meet in person. Google Meet allows us to meet virtually face to face when we can't meet in person. It is also good in the event that we have to have an unplanned meeting quickly and on short notice.

Logical Flow Diagram



Game Mechanics and Systems

Player Character Mechanics

Player Movement:

Risk: Medium

For this game the player must use a Wii balance board to fully control the motorcycle in the game. To control the motorcycle the player must sit or stand on the balance board and lean in the direction they wish to go. If they lean right or left the motorcycle will turn in that direction. Leaning forward or backwards makes the motorcycle speed up or slow down. The player must master these controls to be able to move around the track as quickly and efficiently as possible.

Laps:

Risk: Medium

The goal of this game is for the player to try and get around the track as fast as possible three times. Once they complete the three laps their final time will be displayed and the player will be able to see how they did. So each time the player crosses the finish line they will be one lap closer to completing the game.

Timer:

Risk: Low

Once the game starts and the player begins racing around the track a timer will start to give the player an idea of how long they have been racing for. Once the player completes lap three their time will be displayed so that they can see how fast they were able to complete the game. There will be no additional effects to the timer, it will run continuously and at a constant speed until the player finishes the game, and there are no negatives for getting a “bad time”.

Trail:

Risk: Medium

As the player races around the track a trail will emerge from behind the motorcycle and be horizontal to the track surface. This trail will remain on the track for the duration of the game and will give the player a speed boost while they are driving on it. So after the first lap if the player follows the same path as their last lap they will have the ability to go faster and get a quicker time overall.

Course:

Risk: Low

The track that the player will be racing around will have a mixture of straight aways and turns to have some variety to the track. In the straights the player will be able to go fast and not have to worry about going off the track. The turns will be a mixture of tight and slow and long and fast, that way the player is challenged to use the balance board to maneuver through the turns. So with the above components the player will be able to both go fast and have to concentrate making them feel rewarded once they complete the three laps.

Changing Visuals:

Risk: Low

As the player is driving around the track there will be a camera blurring effect which will be active when the player is going fast. The faster the player goes the more the

background will blur as to recreate the effect of driving fast in real life. This effect will only be present on the sides of the camera and not too close to the center so as to not obstruct the player's view of the track.

Check Points:

Risk: Low

To make sure the player actually completes a lap around the track, there will be “checkpoints” around the track which the player must all hit in order to have the lap count. This stops the player from cheesing the game and just driving back and forth over the finish line. So a system to manage what checkpoints have been crossed by the player will be made so that the game knows if the player has completed a full lap around the track.

UI Systems:

Start Menu:

Risk: Low

A simple menu that the game opens to, to allow the player to start or quit the game.

End Menu

Risk: Low

The menu that appears after the player completes the three laps. It contains some information like how quickly the player completed the laps and possibly how quickly previous people's laps were as well as other details if there is time in the project, like how many times walls were hit, average speed, etc. The player will also be able to choose to exit to the main menu, or play again.

Development Pipelines

Design:

Public Variables:

Player Movement

- rotationForce
- maxSpeed
- brakeForce
- maxBackUpSpeed

1. Designers create documents like the VDD and VDG which state information about features and systems in the game, like how they will work.
2. The programmers then work on implementing the features laid out by the designer and the documents using feature branching.
3. The programmers will implement many of the features with public variables which can be changed easily in the future.
4. The designers make level design documentation about how the track in game will be laid out and place it in our shared google doc and in the documents folder in the repository.
5. The designers will then work in the engine to build the track and use tiling assets created by the artist.
6. Then the game will be brought to testing to test mechanics and level layout.
7. Revisions to the game will be made based on the feedback from testing.

8. As art and sound assets are created by the artist and sound designers the designers will place them into the game.

Art:

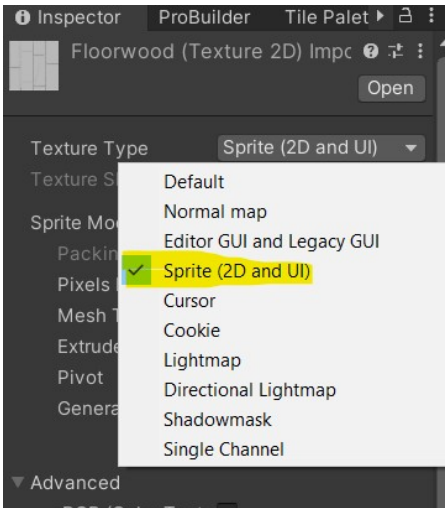
File naming convention: assetType_assetName

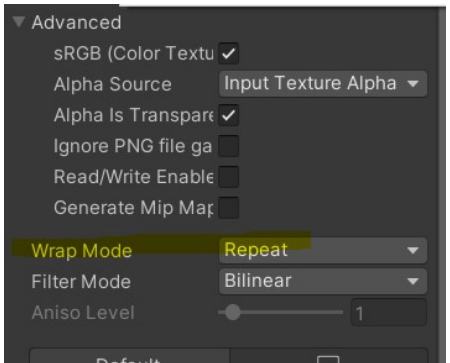
Creating Assets:

1. Artist creates an art asset list for all art assets that will be needed in the game.
2. The artist goes down the list creating each asset over time.
3. The assets are sketched in Photoshop and then shown to the team to get feedback.
4. Asset is then finalized by the artist based on feedback.
5. Asset is then saved as a PNG using the proper naming convention.
6. Artist creates a branch in the repository for implementing art and places the asset into the Unity art asset folder.
7. Asset is then placed into a testing scene so sorting layers can be given to it.
8. Artist then makes the asset a prefab so that it can be easily used and placed into the environment by the designer.
9. Changes are then committed and pushed to the art implementation branch and the team is notified by the artist of the work they have done.
10. Branch is then merged with main and pushed.

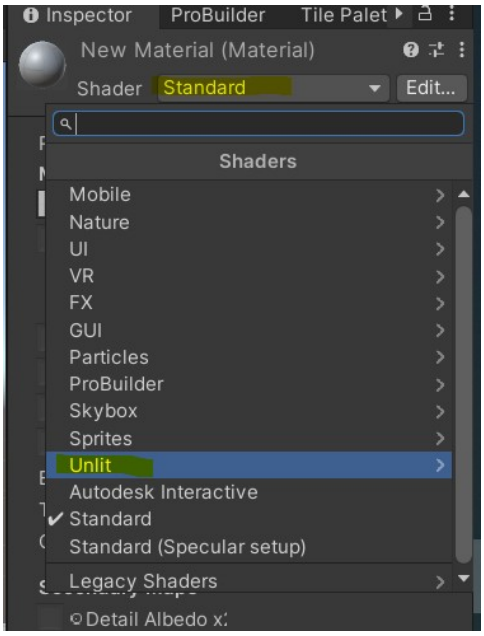
Creating Materials:

1. For art assets that need to be a tile set they will be added to the project like usual.
2. Then the artist will edit the texture settings to set the texture type to “Sprite (2D and UI)”.
3. Then the wrap mode is set to “Repeat”.





4. Finally the artist creates a material in the materials folder and sets the shader to



   “Unlit > texture”.
5. Then the tile can be used by the designers and placed into the game when it is needed.

Audio:

File naming convention: [type]_ [NameOfSound] _[version]

1. A feature is decided to be implemented that requires a sound effect.
2. Sound designer creates documents for what features will have sound effects and what they will be like.
3. Sound designer uses OP-1 and Ableton Live 10 to make some sounds and sound effects.
4. If realistic sounds are needed like the motorcycle, then the sound designer will record sound clips directly and then use Ableton Live 10 to edit those sounds.
5. For composing music Logic will be used by the sound designer.

6. Once a sound asset is fully edited, sound designer exports assets as an mp3 and places them in the project to be used once the feature/s is/are implemented.
7. Once the features are completed by the programmer audio sources are added to the objects which need to play sounds and the programmer makes scripts to play the sounds on specific triggers.

Sprint Updates

Sprint 1:

For this first sprint our main focus was on coming up with the concept for our game. We decided on Speedy Speed Bike which is a motorcycle racing game using a Wii balance board as a controller. Once we had decided on a concept we got to work on making the required documents for the first sprint, getting the repository working and getting an early build. The build was very simple with just a cube for the player which moves forward and backward and turns left and right. For now we just were using the keyboard to control it as we were working on getting our hands on an actual balance board. So by the end of the sprint we had our documents, a build and access to a balance board so we are in a good place for the following sprints.

Sprint 2:

For this second sprint we mainly focused on getting the Wii balance board to work. We were able to do that and control the player character with the board. Another goal for this sprint was to have all player actions, (moving and rotating) done with forces. The values for the different player actions were also tweaked so that the controls with the board feel good. The application that connects the balance board to the computer starts when the build starts and console logs now write to .txt files. After this sprint we are in a good place to test the build and get initial feedback on how the movement in the game.