Attack of the Beat

Project Members: Sean McManes
Advisor: Professor Chia Han
April 17, 2020

Table of Contents

Project Description

The goal of this project is to create a game that is an enjoyable experience for the player by challenging them, creating an interesting game environment, and has game mechanics that have not been fully explored by other games.

The main aspect that would make this game stand out from others is the use of game music that would manipulate the gameplay and environment. This would make the player not only have to look for visual cues in the game, but also audio cues in the music in order to have an advantage and complete the game.

Current games typically use the game music to set the tone of the world/gameplay. For example, creating a suspenseful setting for a scary/horror game with low to no sound, or playing exciting music in an action filled game to get the player immersed in that game. Some games have adapted their gameplay to the music or vice versa, but very few have directly used the music in order to modify the game experience. The ones closest to doing this have stopped short and only do basic implementations, such as movement or attacks based on a constant beat of a generic song.

For years, we have been invested in the game industry and have had experience with several game genres. Most of the enjoyment we found in these games stem from new experiences that the game present to the player. These experiences include new and interesting game mechanics or new interpretations to prior game mechanics, interesting world design, natural game controls and visuals, and more.

Our approach to this problem would include the following:

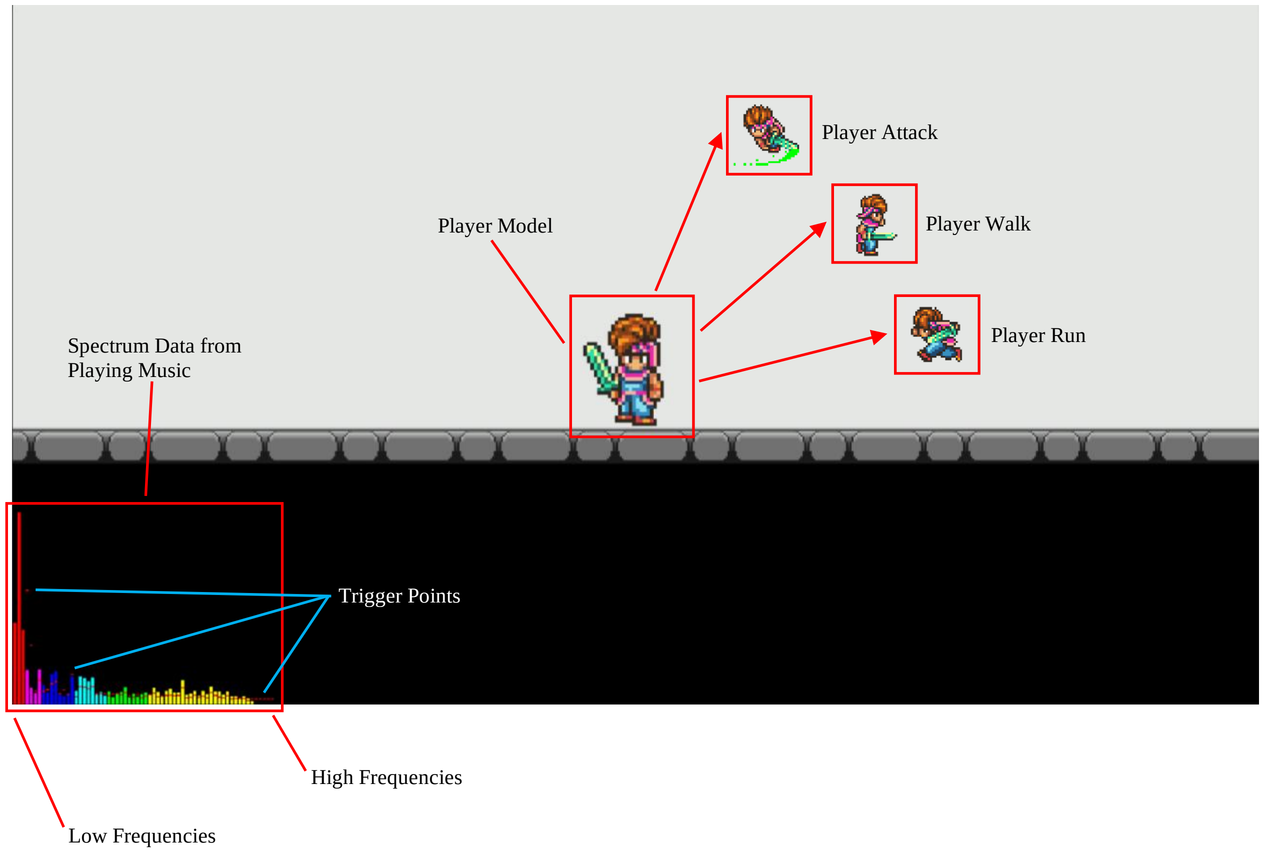
- Creating a tool/library that can analyze the game music that is being played and output the data retrieved to the running game or store it to be used when needed
  - The data stored would most likely be the frequencies found at various times in the music
- Create a working game environment that has an interesting design, interactable objects or items for the player, non-playable characters, and more to fill this environment
- With the analyzed music and working game environment, then aspects of the game can be manipulated by the data found. Some of these aspects include:
  - Enemy/game actions, such as having enemy attacks based on the frequencies of the current music
    - The type of attacks, power, and other aspects can be affected by this as well
  - Game pace can be affected by the type of music being played, with slower song making the game pace slower and fast, high energy, songs would make the pace faster and perhaps make the game harder

Aspects of the world design could also be affected by the song, with lighting and room design changing with the music

Abstract

Current games use several aspects to immerse the player and make enjoyable experiences, but they do not use the game music beyond atmosphere. This project aimed to make a game with mechanics manipulated by music so the player must listen to the music along with visual cues to succeed. This was accomplished with object manipulation to visually mimic data acquired from continuous music analysis.

User Interface Specification



Test Plan: ***Part I - Description of Overall Test Plan***

The approach taken to test this project will take place on an individual parts basis and then an overall complete project basis. Throughout each test, particularly with the individual parts, depending on what results need to be examined, the project will produce data necessary for the tests. This data will be analyzed and compared to the intended results, and from that conclusions will be drawn, and further improvements/changes will be determined from the findings. After these parts are working as intended, they will be put together towards the overall project and then final tests can be conducted to make sure that the parts work together as intended, and the overall project is working as intended. Any improvements/changes would be made if needed depending on the outcomes.

<u>Test Plan: *Part II - Test Case Descriptions*</u>

1.1 Game Map and Texture Test 1
1.2 This test ensures that the first round of map textures and map generators work as intended
1.3 This test will run the map generator with the first round of textures to show that the generator is picking the correct textures for the given test cases, and that the textures are correctly designed to work with intended situations.
1.4 Inputs: First round textures and test map layout
1.5 Outputs: Map made from textures given based on map layout
1.6 Normal
1.7 Whitebox
1.8 Functional
1.9 Unit Test

2.1 Player Texture and Animation Test 1
2.2 This test ensures that the initial player textures and animations work as intended
2.3 This test will have the player character complete various test actions to show that the textures flow together to make the animations look correct. This will also test the timing between each part of the animation, and add/remove time if needed
2.4 Inputs: Player textures and test actions
2.5 Output: Player animation matching the given actions
2.6 Normal
2.7 Whitebox
2.8 Functional
2.9 Unit

3.1 Game Map and Texture Test 2
3.2 This test ensures that the game map generator handles exceptions as intended
3.3 This test will run the map generator with the first round of textures to show that the generator correctly identifies errors in the given map design, displays proper warnings, and displays proper textures in the areas that have errors.
3.4 Inputs: First round textures and test map layout with errors
3.5 Output: Map made from textures given based on map layout with proper error messages and error textures
3.6 Abnormal
3.7 Whitebox
3.8 Functional
3.9 Unit

4.1 Music Analyzer Test 1
4.2 Run the music analyzer over test music
4.3 This test will run the music analyzer over the first set of test music and see if the output from the analyzer match what was expected. The major part of this is making sure that the analyzer is picking good points in the music for specific actions, and any issues would be adjusted.
4.4 Inputs: Test music (Set 1)
4.5 Outputs: Action codes with timestamps from trigger points
4.6 Normal
4.7 Whitebox
4.8 Functional
4.9 Unit

5.1 Music Analyzer Test 2
5.2 Run the music analyzer after modifications over different test music
5.3 This test will run the music analyzer over another set of test music and see if the output from the analyzer match what was expected. This is to see if the adjustments made from the prior test cause incorrect outputs from other songs, and depending on the outcome, some tweaks to the design may be done. Generally, the type of music will be the same.
5.4 Inputs: Test music (Set 2)
5.5 Outputs: Action codes with timestamps from trigger points
5.6 Normal
5.7 Whitebox
5.8 Functional
5.9 Unit

6.1 Alternative Music Analyzer Test
6.2 Run the music analyzer over different genre test music
6.3 This test will run the analyzer over another set of test music from a different genre from the music from the other tests. This is to see if the settings made in the prior tests work well with other genres of music, and if not, then what adjustments can be made to get the analyzer working
6.4 Inputs: Alternative test music
6.5 Outputs: Action codes with timestamps from trigger points
6.6 Normal
6.7 Whitebox
6.8 Functional
6.9 Unit

7.1 Player/Object Movement and Actions Test 1
7.2 Test the various movements and actions of the player character and other objects
7.3 This test will ensure that the player movement and actions are working as expected and that there are no disconnect or lag with them. This test will also allow us to adjust settings, such as how far characters move per frame and so on.
7.4 Inputs: Test user input (movements and actions)
7.5 Outputs: Displaying player movements and actions
7.6 Normal
7.7 Whitebox
7.8 Functional
7.9 Unit

8.1 Gameplay and Music General Test 1
8.2 Test the game loop with music implementation
8.3 This test will ensure that the music playing with the game is outputting the action codes, as tested before (as a separate part), at the correct times and does not run into issues over time. Major points for investigation are desync from the game music, stuttering, incorrect actions, and more.
8.4 Inputs: Test music and generic game inputs
8.5 Outputs: Game display, music, and corresponding action codes
8.6 Normal
8.7 Whitebox
8.8 Functional
8.9 Unit

9.1 Player/Object Movement and Actions Test 2
9.2 Test the various movements and actions of the player character and other objects with music
9.3 This test will ensure that the music, action points gained from the music, and object movements from these action points are working in sync and as expected. Any issues that arise with the implementation of these action points and actual object movement can be addressed at this point.
9.4 Inputs: Test music
9.5 Outputs: Displaying object movements and actions, music, and action codes with timestamps
9.6 Normal
9.7 Whitebox
9.8 Functional
9.9 Unit

10.1 Gameplay and Music General Test 2
10.2 Test the game loop with music implementation and actions from the music
10.3 This test will ensure that the music playing with the game is outputting the action codes, as tested before (as a separate part), at the correct times and does not run into issues over time. This will add a player-controlled character, and several computer/music controlled characters to mainly stress test the game and make sure that these elements do not create conflicts.
10.4 Inputs: Test music and player test input
10.5 Outputs: Game display, music, and corresponding action codes
10.6 Normal
10.7 Whitebox
10.8 Performance
10.9 Unit

<u>Test Case: *Part III - Test Case Matrix*</u>

	Normal/Abnormal	**Blackbox/Whitebox**	**Functional/Performance**	**Unit/Integration**
1	Normal	Whitebox	Functional	Unit
2	Normal	Whitebox	Functional	Unit
3	Abnormal	Whitebox	Functional	Unit
4	Normal	Whitebox	Functional	Unit
5	Normal	Whitebox	Functional	Unit
6	Normal	Whitebox	Functional	Unit
7	Normal	Whitebox	Functional	Unit
8	Normal	Whitebox	Functional	Unit
9	Normal	Whitebox	Functional	Unit
10	Normal	Whitebox	Performance	Unit

Test Plan: ***Results***

1. First round of textures were the wall textures. During the first runs, there were minor errors with corners using incorrect textures. After testing various cases, the issue in logic was found in the map generator and was corrected. Additional runs of these tests showed no further issues.

2. The player textures during the first run immediately had issues. When the player would rotate or change to other sets of animations it seemed to shift from its "center position." After reviewing the images used for the animations, it was found that there were inconsistencies with the size of each set of images. As such, when changing animations, the animator would change center and would not always match the new image set. All images were put into a photo editor, and a global center was established, and all images were aligned, and then cropped to an equivalent size for each image. After these images were used as replacements, the issues were resolved, and no further issues occurred.

3. Tested the map creator with intentional errors in the map design. Specifically having an incorrect wall width and length that would be too small for the texture designed for the game. In these cases, the game properly built the game, but used a place holder texture, that would clearly indicate to the developer that the map design has an error, and a message properly displayed what the error what and where on the map it was located.

4. This test used some simple music that played sets of various frequencies so that it would be clear if the visual frequency graph in the game displayed the proper information corresponding to the music. Along with this, the analyzer began to choose actions based on these frequencies, but some actions were happening too often / not enough, so various adjustments were made to get the analyzer working as intended. After several rounds, this goal was met.

5. The test used a similar genre of music, but more complex than the prior test. Again, some actions happen too often and others not enough, so adjustments were made, and additional functions for action detection. Specifically, additions that created adapting trigger points along the spectrum that would increase in difficulty or decrease based on the number of times the prior triggers were hit.

6. This test used a different genre of music than the prior tests. After the prior adjustments, they worked well with the different genre, and very few fine adjustments were made to the existing changes from the last test that would suit each genre.

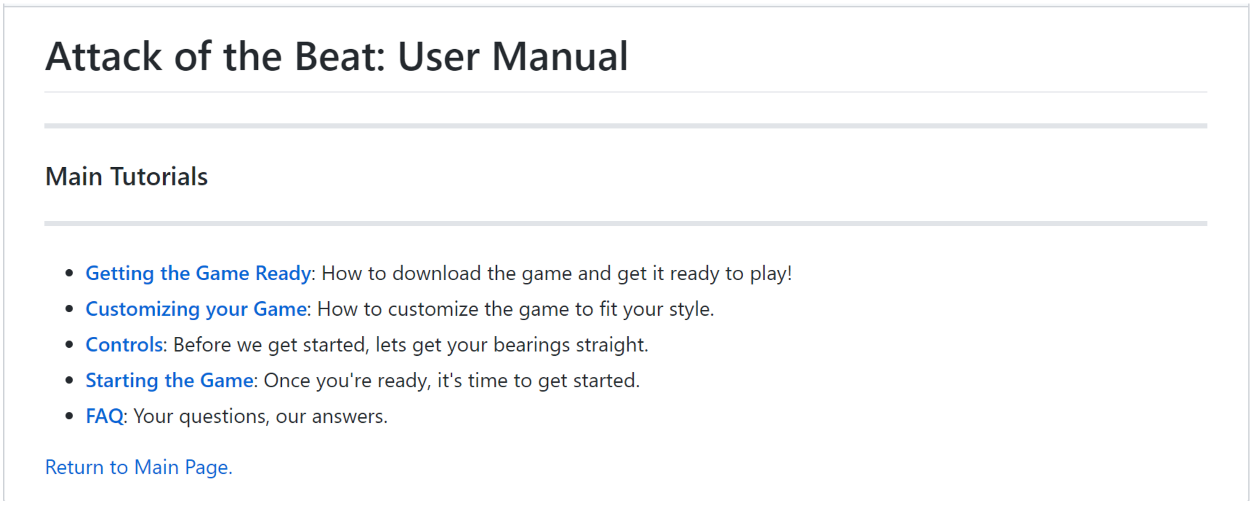
7. The attack actions and movement of the player character were tested in various situations both alone and with other objects. Alone, the attacks animated properly and had no animation issues and the player movement speed seemed to match the corresponding animation. With other objects that it could collide with, certain situations would result in the colliders becoming overlapped and could force the character into an out-of-bounds area. The types of colliders used and where they were positioned on the character were adjusted to make the game less susceptible to these issues. Character attacks were changed so that the weapon would not collide with the walls. This could easily be adjusted later, but when the attack ignored the wall, the gameplay seemed to look better, as before the contact would result in unwanted player movement.

8. This test used the music tested before in order to test the logic and actions of the laser spawners in the game that are manipulated by this music. Initially the lasers that spawned, which are layered and colored separately based on the frequencies they are affected by, began to flash repeatedly and their collision and length detection was not properly working. This is in place so that when the laser collides with a wall or player, it does not continue to go through the object, but stops at the point of collision. After review and further rounds, the issue was resolved by adjusting the raycast that was used to find collisions with the laser. It was colliding with the lasers that were drawn the frame before, and its rotation did not match that of the laser spawner. Once the rotations were aligned and the lasers were excluded from the collisions, the laser acted as intended, and stopped expanding on contact with objects. Even with these additions, the music continued to stay in sync and no game slowdowns were noticed during the test.

9. After the adjustments from the last test, the lasers worked properly with collision detection, but the length that they went to for some frequencies were too long. Some resulted in lasers going across the whole map (if collisions were off). So, adjustments were made to make the laser length closer to design and made it so that lower frequencies would have a larger max length, and higher frequencies would have a smaller max length. With this, the collisions continued to work as intended and the lasers acted as intended.

10. This stress test ran for an extended period with several music affected objects and multiple test songs in order to see if there was any slowdown, crashes, or other general errors. Due to the changes in the prior tests, there were no noticeable errors or crashes during this test.

User Manual: *Main Tutorials*



Attack of the Beat: User Manual

Main Tutorials

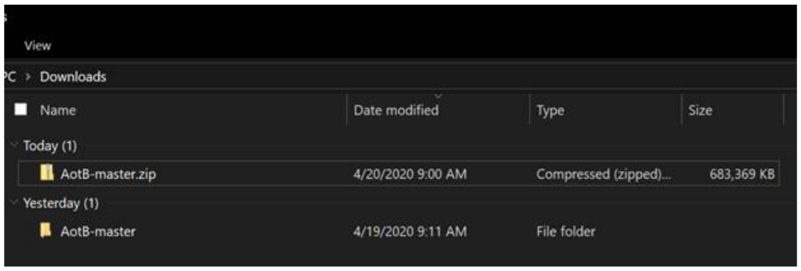
- Getting the Game Ready: How to download the game and get it ready to play!
- Customizing your Game: How to customize the game to fit your style.
- Controls: Before we get started, lets get your bearings straight.
- Starting the Game: Once you're ready, it's time to get started.
- FAQ: Your questions, our answers.

Return to Main Page.

User Manual: *Getting the Game Ready*

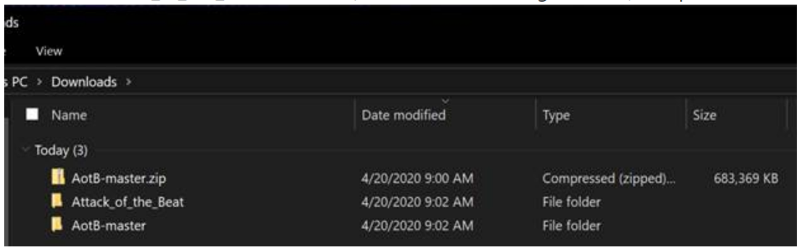
Getting the Game Ready

- Download/Clone Attack of the Beat



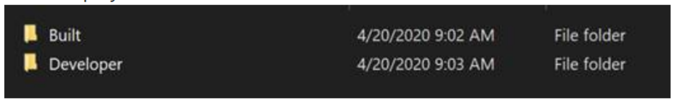
Name	Date modified	Type	Size
Today (1)			
AotB-master.zip	4/20/2020 9:00 AM	Compressed (zipped)...	683,369 KB
Yesterday (1)			
AotB-master	4/19/2020 9:11 AM	File folder	

- Move the Attack_of_the_Beat subfolder, which contains the game files, to a preferred folder location.



Name	Date modified	Type	Size
Today (3)			
AotB-master.zip	4/20/2020 9:00 AM	Compressed (zipped)...	683,369 KB
Attack_of_the_Beat	4/20/2020 9:02 AM	File folder	
AotB-master	4/20/2020 9:02 AM	File folder	

- Our recommendation is to keep everything under the Attack_of_the_Beat folder, which will then have two versions:
  - Built - this version can be played from a single executable
  - Developer - this version allows you to edit game files, but requires the Unity Editor and you must build the game in order to play

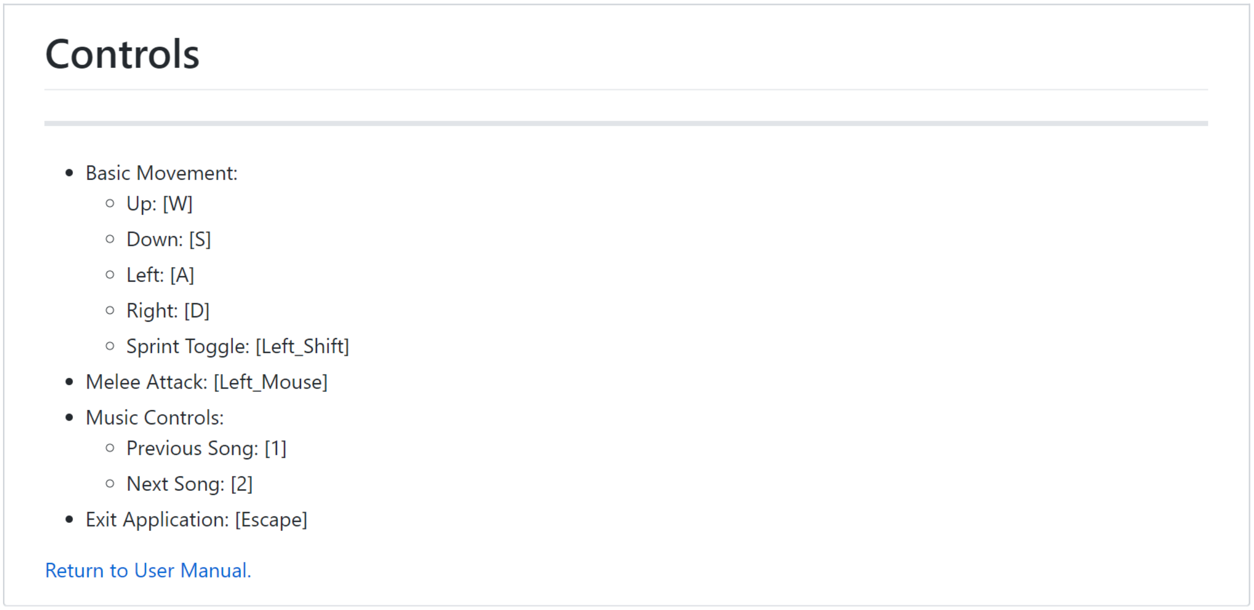


Built	4/20/2020 9:02 AM	File folder
Developer	4/20/2020 9:03 AM	File folder

- For more details on starting either version, see Starting the Game.

Return to User Manual.

User Manual: *Controls*



Controls

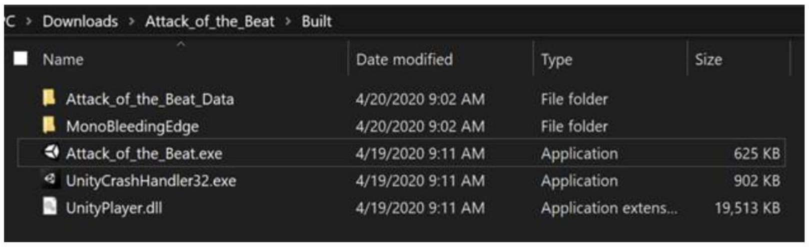
- Basic Movement:
    - Up: [W]
    - Down: [S]
    - Left: [A]
    - Right: [D]
    - Sprint Toggle: [Left_Shift]
- Melee Attack: [Left_Mouse]
- Music Controls:
    - Previous Song: [1]
    - Next Song: [2]
- Exit Application: [Escape]

Return to User Manual.

User Manual: *Starting the Game*

Starting the Game

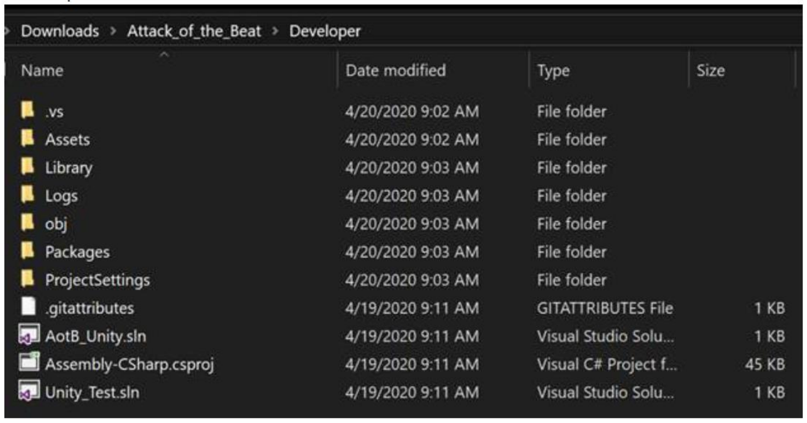
- Pre-Built Game Version
  - Run Attack_of_the_Beat.exe
    - Located user_path/Attack_of_the_Beat/PreBuilt/Attack_of_the_Beat.exe
    - Built Folder:



C > Downloads > Attack_of_the_Beat > Built

Name	Date modified	Type	Size
Attack_of_the_Beat_Data	4/20/2020 9:02 AM	File folder	
MonoBleedingEdge	4/20/2020 9:02 AM	File folder	
Attack_of_the_Beat.exe	4/19/2020 9:11 AM	Application	625 KB
UnityCrashHandler32.exe	4/19/2020 9:11 AM	Application	902 KB
UnityPlayer.dll	4/19/2020 9:11 AM	Application extens...	19,513 KB

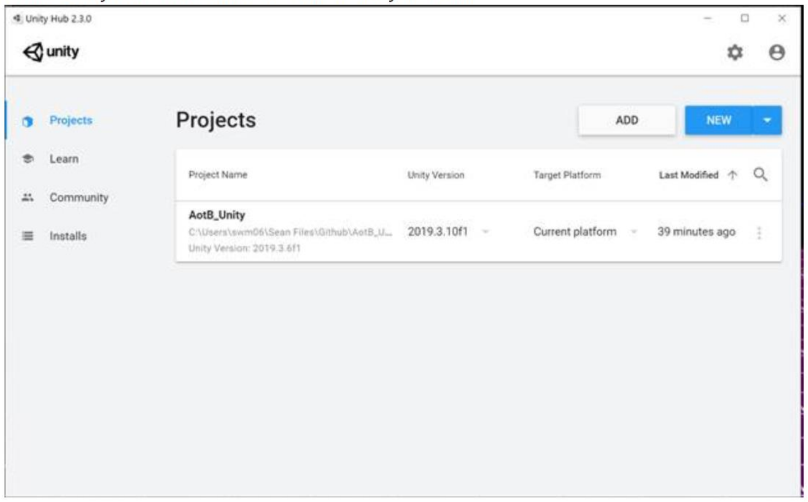
- Developer Version
  - Developer Folder:



Downloads > Attack_of_the_Beat > Developer

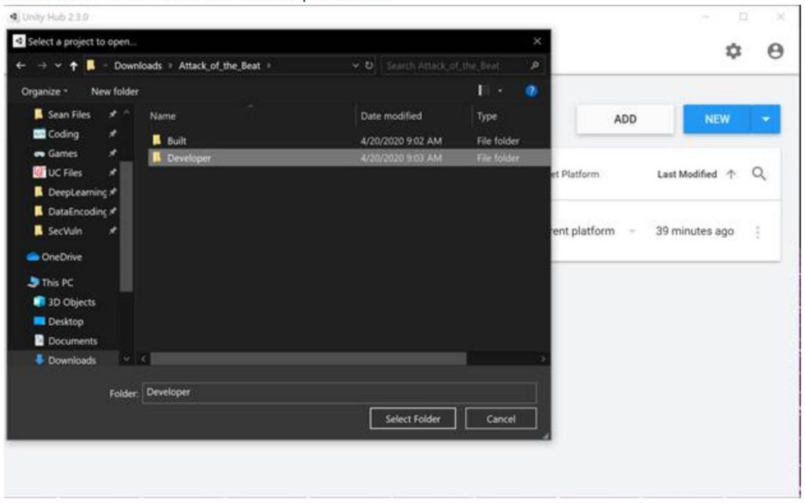
Name	Date modified	Type	Size
.vs	4/20/2020 9:02 AM	File folder	
Assets	4/20/2020 9:02 AM	File folder	
Library	4/20/2020 9:03 AM	File folder	
Logs	4/20/2020 9:03 AM	File folder	
obj	4/20/2020 9:03 AM	File folder	
Packages	4/20/2020 9:03 AM	File folder	
ProjectSettings	4/20/2020 9:03 AM	File folder	
.gitattributes	4/19/2020 9:11 AM	GITATTRIBUTES File	1 KB
AotB_Unity.sln	4/19/2020 9:11 AM	Visual Studio Solu...	1 KB
Assembly-CSharp.csproj	4/19/2020 9:11 AM	Visual C# Project f...	45 KB
Unity_Test.sln	4/19/2020 9:11 AM	Visual Studio Solu...	1 KB

- Install Unity Hub and Install the Latest Unity Version.



Project Name	Unity Version	Target Platform	Last Modified
AotB_Unity C:\Users\swm06\Sean Files\Github\AotB_U... Unity Version: 2019.3.6f1	2019.3.10f1	Current platform	39 minutes ago

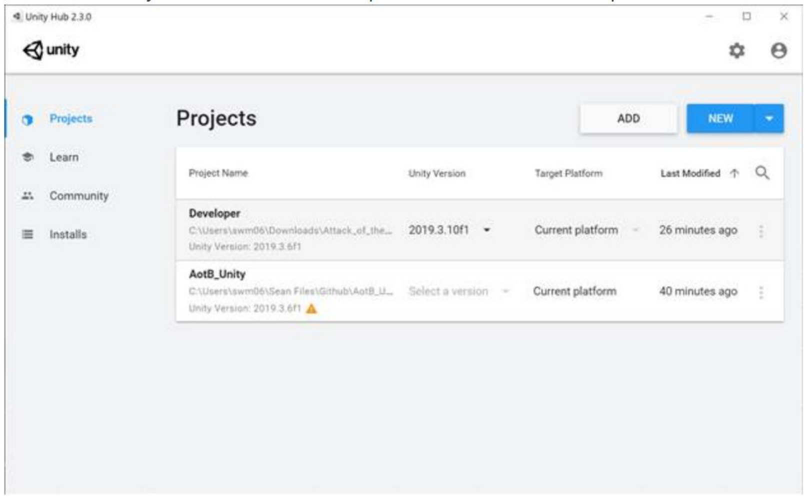
- Select Add and Select the Developer Folder



Name	Date modified	Type
Built	4/20/2020 9:02 AM	File folder
Developer	4/20/2020 9:03 AM	File folder

Folder: Developer

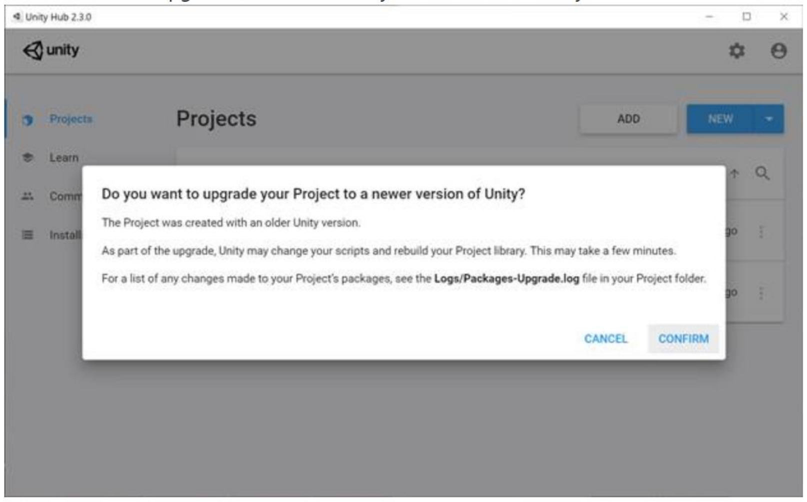
- Select Your Unity Version Next to Developer and Launch the Developer Files



Projects

Project Name	Unity Version	Target Platform	Last Modified
Developer C:\Users\swm06\Downloads\Attack_of_the... Unity Version: 2019.3.6f1	2019.3.10f1	Current platform	26 minutes ago
AotB_Unity C:\Users\swm06\Sean Files\Github\AotB_U... Unity Version: 2019.3.6f1	Select a version	Current platform	40 minutes ago

- Click Confirm to Upgrade the Game to Any New Version of Unity



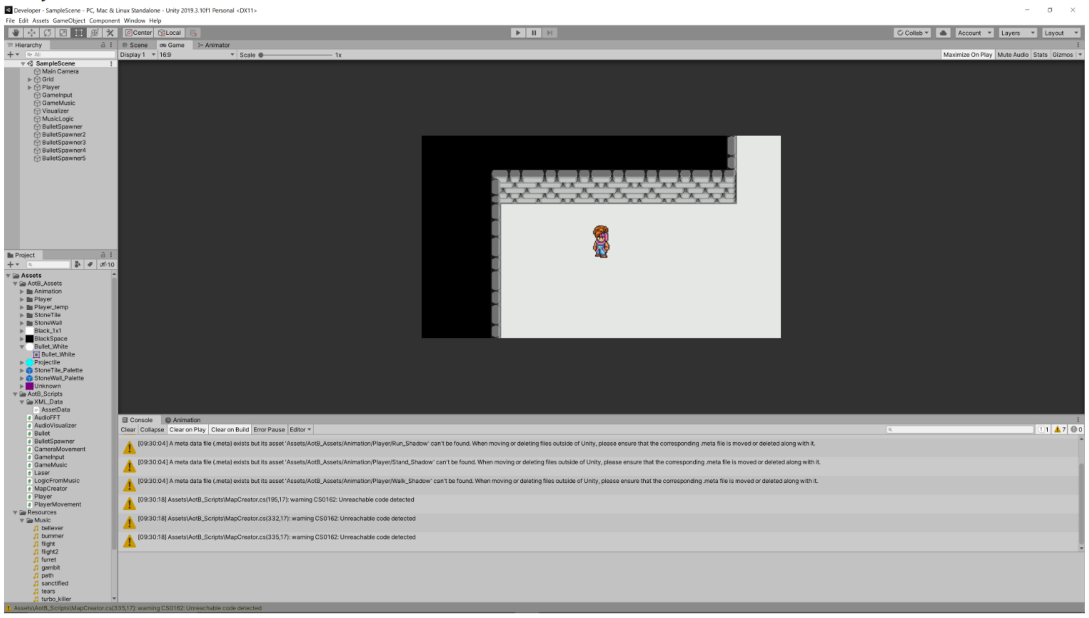
Do you want to upgrade your Project to a newer version of Unity?

The Project was created with an older Unity version.

As part of the upgrade, Unity may change your scripts and rebuild your Project library. This may take a few minutes.

For a list of any changes made to your Project's packages, see the **Logs/Packages-Upgrade.log** file in your Project folder.

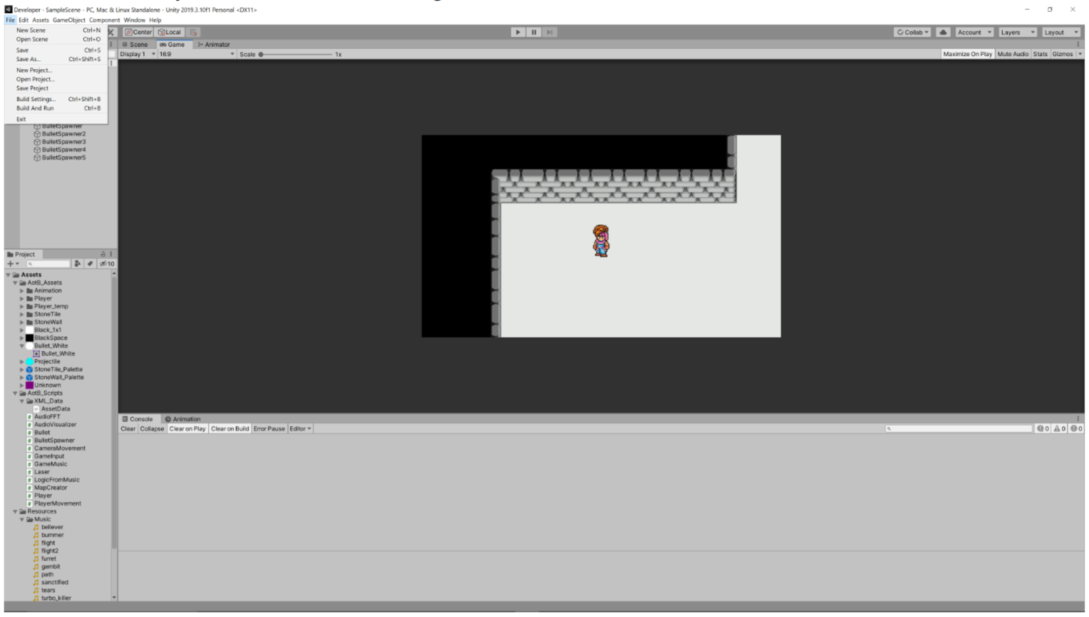
- Unity Editor Will Launch and You Should See This Screen



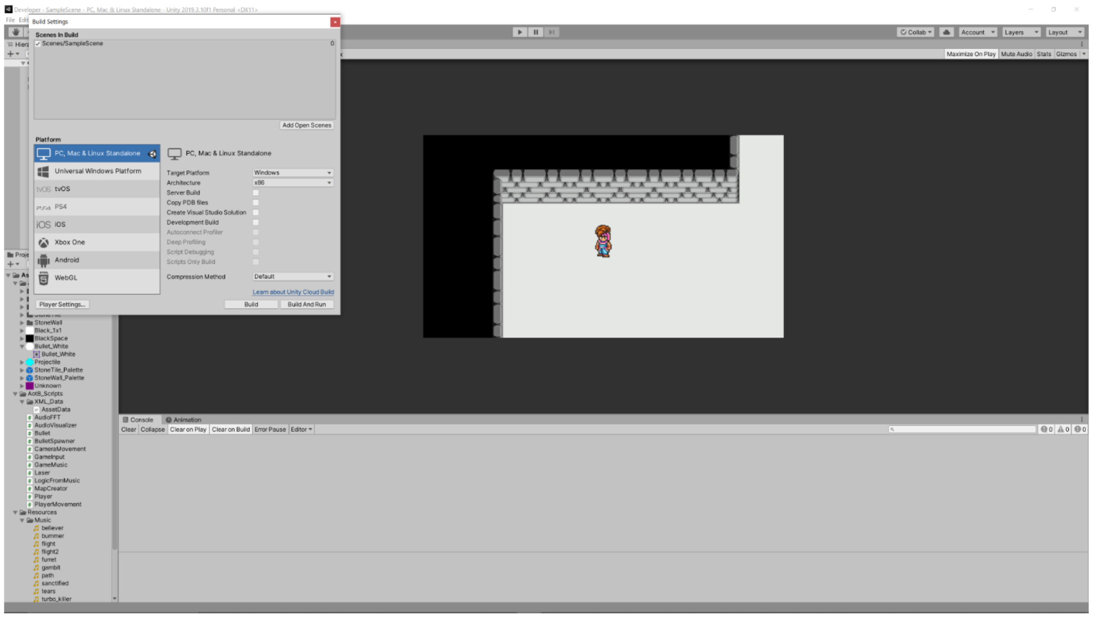
```
[09:30:04] A meta data file (.meta) exists but its asset 'Assets/AotB_Assets/Animation/Player/Run_Shadow' can't be found. When moving or deleting files outside of Unity, please ensure that the corresponding .meta file is moved or deleted along with it.
[09:30:04] A meta data file (.meta) exists but its asset 'Assets/AotB_Assets/Animation/Player/Stand_Shadow' can't be found. When moving or deleting files outside of Unity, please ensure that the corresponding .meta file is moved or deleted along with it.
[09:30:04] A meta data file (.meta) exists but its asset 'Assets/AotB_Assets/Animation/Player/Walk_Shadow' can't be found. When moving or deleting files outside of Unity, please ensure that the corresponding .meta file is moved or deleted along with it.
[09:30:18] Assets\AotB_Scripts\MapCreator.cs(195,17): warning CS0162: Unreachable code detected
[09:30:18] Assets\AotB_Scripts\MapCreator.cs(332,17): warning CS0162: Unreachable code detected
[09:30:18] Assets\AotB_Scripts\MapCreator.cs(335,17): warning CS0162: Unreachable code detected
```

- See Customizing Your Game to Add Your Own Music

- Once You Are Ready, Select File and Build Settings



- Select Build



Developer - SampleScene - PC, Mac & Linux Standalone - Unity 2019.3.10f1 Personal <DX11>

Build Settings

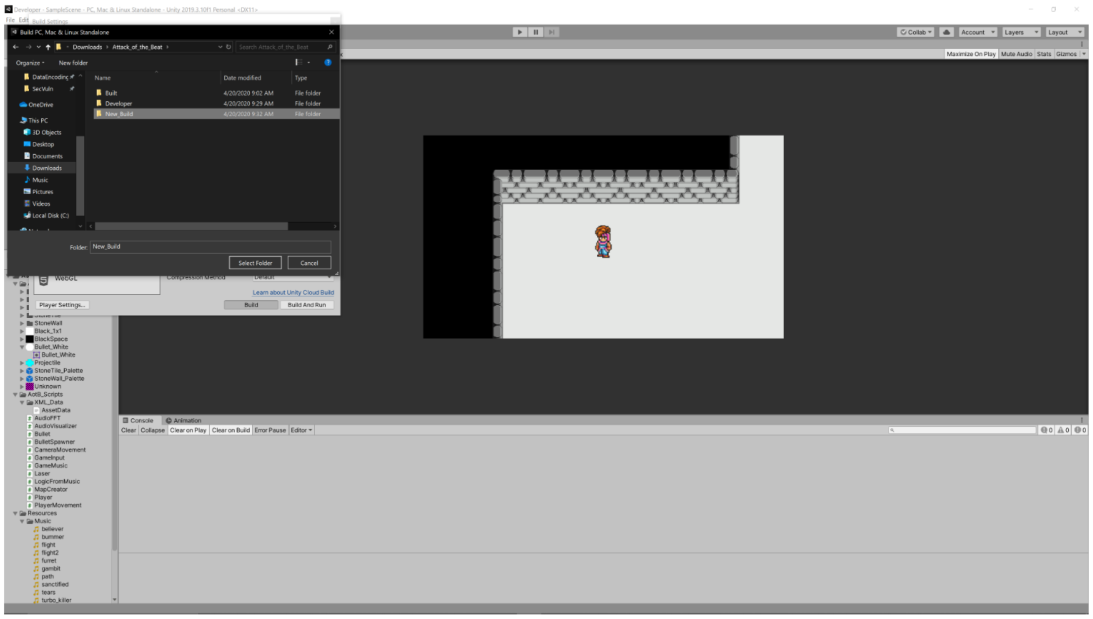
Scenes In Build

✓ Scenes/SampleScene 0

Platform: PC, Mac & Linux Standalone

Target Platform: Windows
Architecture: x86
Compression Method: Default

- Create a New Folder For Your Custom Build and Select That Folder



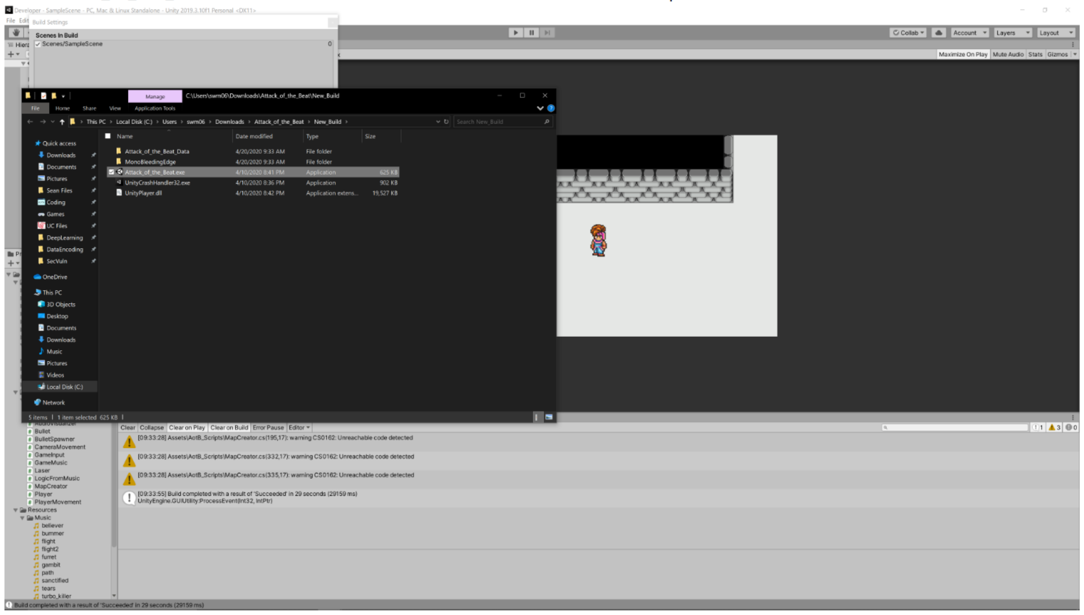
Build PC, Mac & Linux Standalone

Downloads › Attack_of_the_Beat ›

Name	Date modified	Type
Built	4/20/2020 9:02 AM	File folder
Developer	4/20/2020 9:29 AM	File folder
New_Build	4/20/2020 9:32 AM	File folder

Folder: New_Build

- Run Attack_of_the_Beat.exe From Your Custom Build Location Once It Completes.



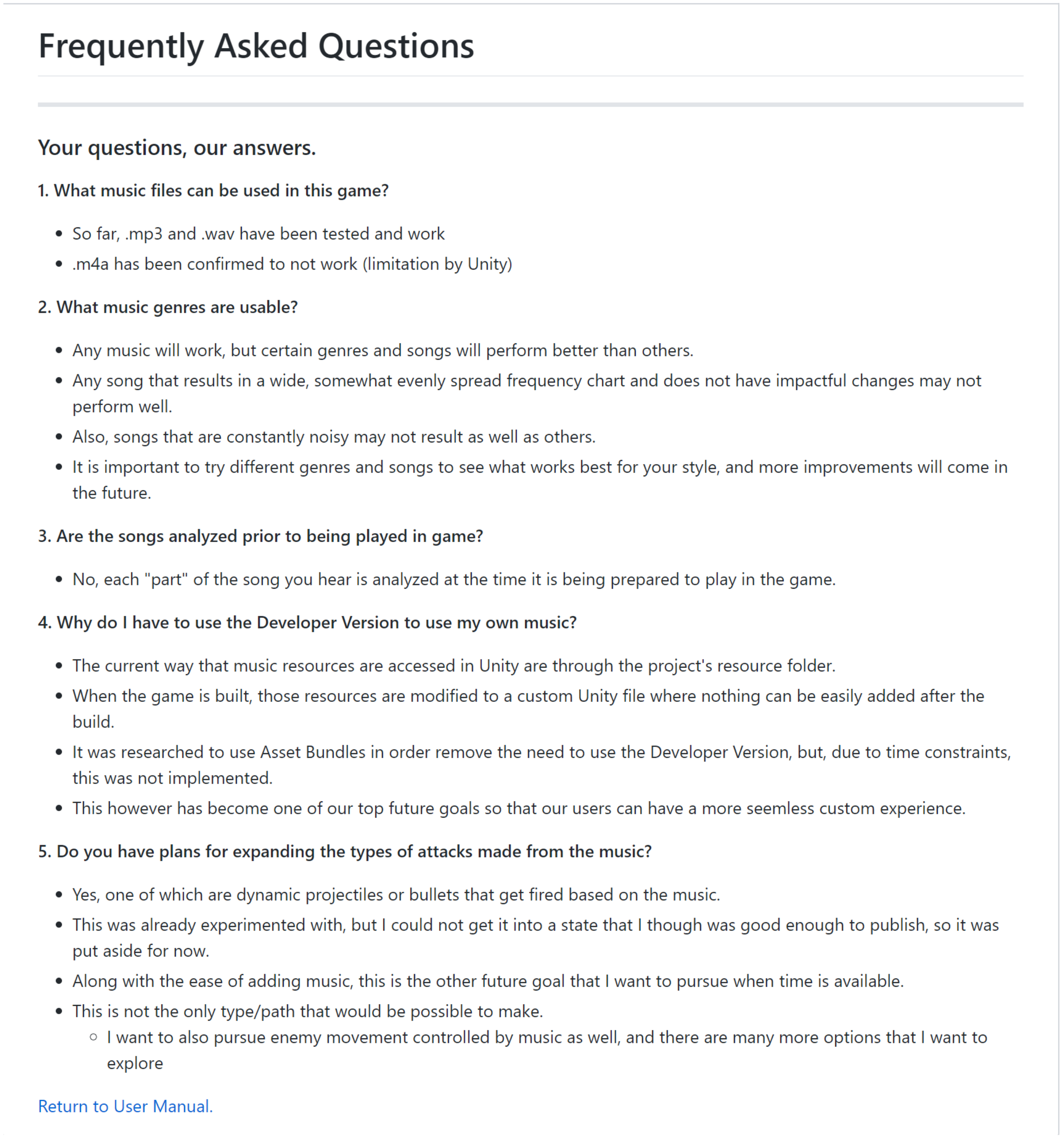
C:\Users\swm06\Downloads\Attack_of_the_Beat\New_Build

Name	Date modified	Type	Size
Attack_of_the_Beat_Data	4/20/2020 9:33 AM	File folder	
MonoBleedingEdge	4/20/2020 9:33 AM	File folder	
Attack_of_the_Beat.exe	4/10/2020 8:41 PM	Application	625 KB
UnityCrashHandler32.exe	4/10/2020 8:36 PM	Application	902 KB
UnityPlayer.dll	4/10/2020 8:42 PM	Application extens...	19,527 KB

```
[09:33:28] Assets\AotB_Scripts\MapCreator.cs(195,17): warning CS0162: Unreachable code detected
[09:33:28] Assets\AotB_Scripts\MapCreator.cs(332,17): warning CS0162: Unreachable code detected
[09:33:28] Assets\AotB_Scripts\MapCreator.cs(335,17): warning CS0162: Unreachable code detected
[09:33:55] Build completed with a result of 'Succeeded' in 29 seconds (29159 ms)
UnityEngine.GUIUtility:ProcessEvent(Int32, IntPtr)
```

Return to User Manual.

User Manual: *Frequently Asked Questions*



Frequently Asked Questions

Your questions, our answers.

1. What music files can be used in this game?

- So far, .mp3 and .wav have been tested and work
- .m4a has been confirmed to not work (limitation by Unity)

2. What music genres are usable?

- Any music will work, but certain genres and songs will perform better than others.
- Any song that results in a wide, somewhat evenly spread frequency chart and does not have impactful changes may not perform well.
- Also, songs that are constantly noisy may not result as well as others.
- It is important to try different genres and songs to see what works best for your style, and more improvements will come in the future.

3. Are the songs analyzed prior to being played in game?

- No, each "part" of the song you hear is analyzed at the time it is being prepared to play in the game.

4. Why do I have to use the Developer Version to use my own music?

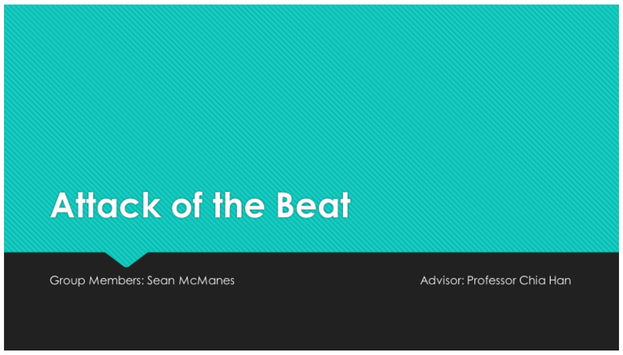
- The current way that music resources are accessed in Unity are through the project's resource folder.
- When the game is built, those resources are modified to a custom Unity file where nothing can be easily added after the build.
- It was researched to use Asset Bundles in order remove the need to use the Developer Version, but, due to time constraints, this was not implemented.
- This however has become one of our top future goals so that our users can have a more seemless custom experience.

5. Do you have plans for expanding the types of attacks made from the music?

- Yes, one of which are dynamic projectiles or bullets that get fired based on the music.
- This was already experimented with, but I could not get it into a state that I though was good enough to publish, so it was put aside for now.
- Along with the ease of adding music, this is the other future goal that I want to pursue when time is available.
- This is not the only type/path that would be possible to make.
  - I want to also pursue enemy movement controlled by music as well, and there are many more options that I want to explore

Return to User Manual.

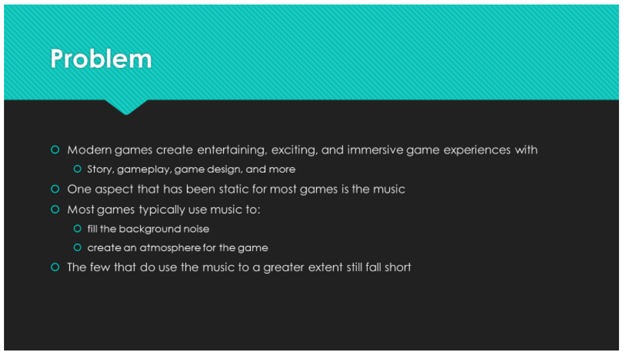
Final PowerPoint Presentation



Attack of the Beat

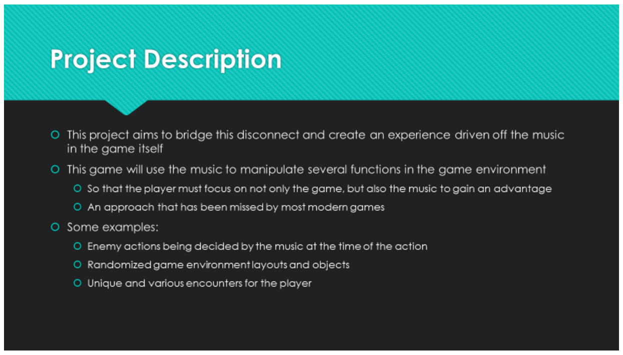
Group Members: Sean McManes

Advisor: Professor Chia Han



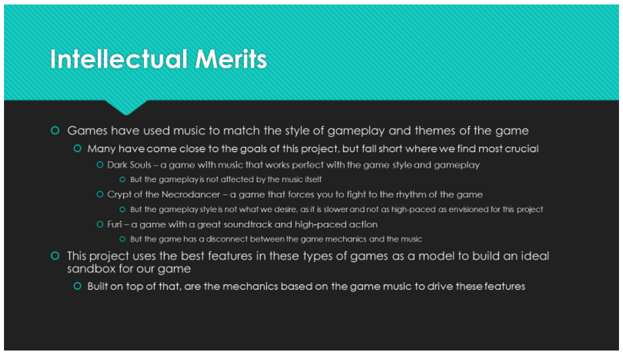
Problem

- Modern games create entertaining, exciting, and immersive game experiences with
  - Story, gameplay, game design, and more
- One aspect that has been static for most games is the music
- Most games typically use music to:
  - fill the background noise
  - create an atmosphere for the game
- The few that do use the music to a greater extent still fall short



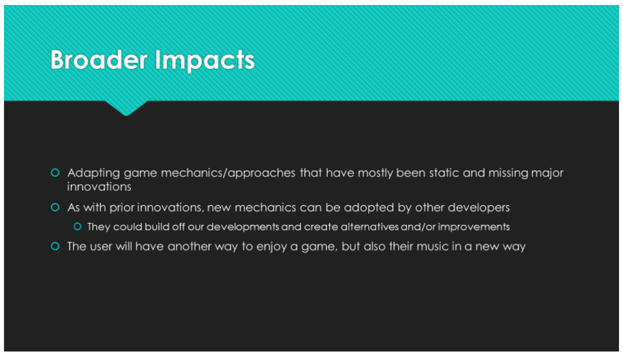
Project Description

- This project aims to bridge this disconnect and create an experience driven off the music in the game itself
- This game will use the music to manipulate several functions in the game environment
  - So that the player must focus on not only the game, but also the music to gain an advantage
  - An approach that has been missed by most modern games
- Some examples:
  - Enemy actions being decided by the music at the time of the action
  - Randomized game environment layouts and objects
  - Unique and various encounters for the player



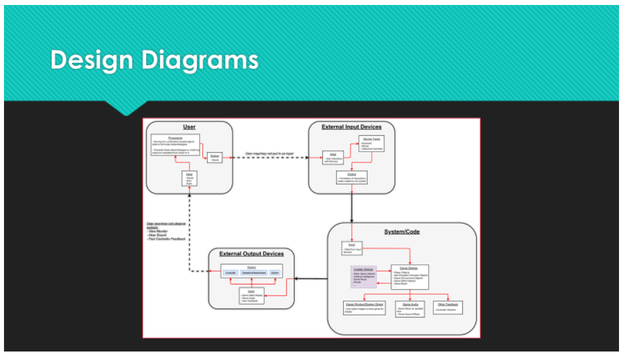
Intellectual Merits

- Games have used music to match the style of gameplay and themes of the game
  - Many have come close to the goals of this project, but fall short where we find most crucial
    - Dark Souls – a game with music that works perfect with the game style and gameplay
      - But the gameplay is not affected by the music itself
    - Crypt of the Necrodancer – a game that forces you to fight to the rhythm of the game
      - But the gameplay style is not what we desire, as it is slower and not as high-paced as envisioned for this project
    - Furi – a game with a great soundtrack and high-paced action
      - But the game has a disconnect between the game mechanics and the music
- This project uses the best features in these types of games as a model to build an ideal sandbox for our game
  - Built on top of that, are the mechanics based on the game music to drive these features

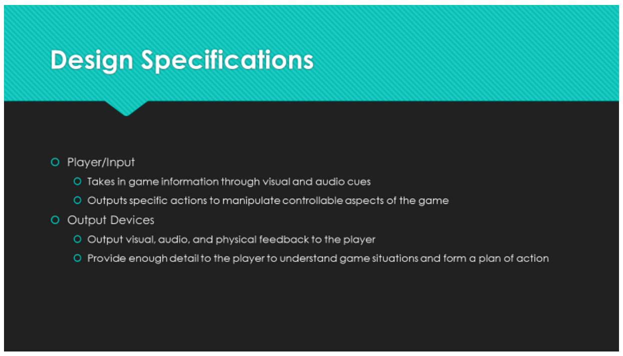


Broader Impacts

- Adapting game mechanics/approaches that have mostly been static and missing major innovations
- As with prior innovations, new mechanics can be adopted by other developers
  - They could build off our developments and create alternatives and/or improvements
- The user will have another way to enjoy a game, but also their music in a new way

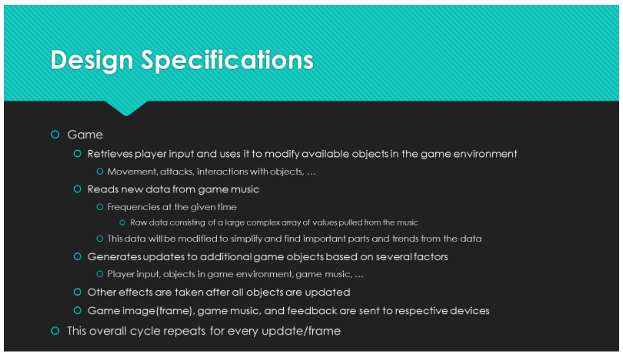


Design Diagrams



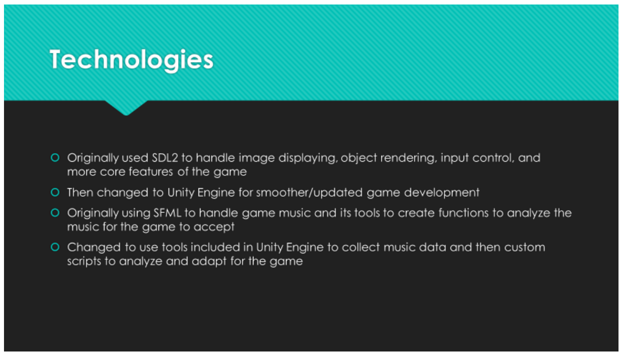
Design Specifications

- Player/Input
  - Takes in game information through visual and audio cues
  - Outputs specific actions to manipulate controllable aspects of the game
- Output Devices
  - Output visual, audio, and physical feedback to the player
  - Provide enough detail to the player to understand game situations and form a plan of action



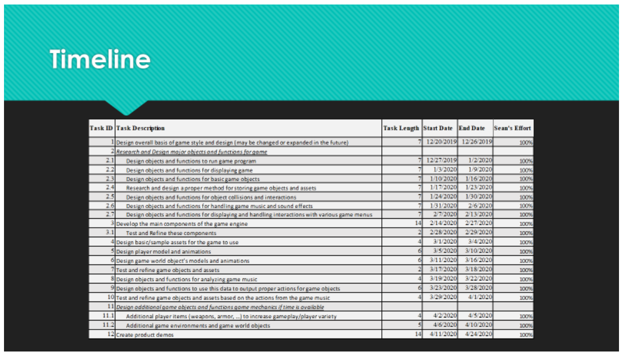
Design Specifications

- Game
  - Retrieves player input and uses it to modify available objects in the game environment
    - Movement, attacks, interactions with objects, …
  - Reads new data from game music
    - Frequencies at the given time
      - Raw data consisting of a large complex array of values pulled from the music
    - This data will be modified to simplify and find important parts and trends from the data
  - Generates updates to additional game objects based on several factors
    - Player input, objects in game environment, game music, …
  - Other effects are taken after all objects are updated
  - Game image(frame), game music, and feedback are sent to respective devices
- This overall cycle repeats for every update/frame



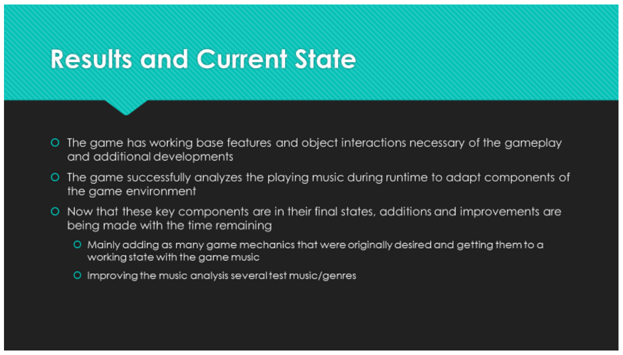
Technologies

- Originally used SDL2 to handle image displaying, object rendering, input control, and more core features of the game
- Then changed to Unity Engine for smoother/updated game development
- Originally using SFML to handle game music and its tools to create functions to analyze the music for the game to accept
- Changed to use tools included in Unity Engine to collect music data and then custom scripts to analyze and adapt for the game



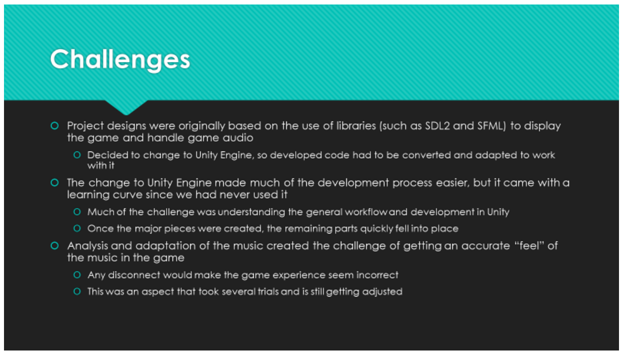
Timeline

Task ID	Task Description	Task Length	Start Date	End Date	Sean's Effort
1	Design overall basis of game style and design (may be changed or expanded in the future)	7	12/20/2019	12/26/2019	100%
2	*Research and Design major objects and functions for game*				
2.1	Design objects and functions to run game program	7	12/27/2019	1/2/2020	100%
2.2	Design objects and functions for displaying game	7	1/3/2020	1/9/2020	100%
2.3	Design objects and functions for basic game objects	7	1/10/2020	1/16/2020	100%
2.4	Research and design a proper method for storing game objects and assets	7	1/17/2020	1/23/2020	100%
2.5	Design objects and functions for object collisions and interactions	7	1/24/2020	1/30/2020	100%
2.6	Design objects and functions for handling game music and sound effects	7	1/31/2020	2/6/2020	100%
2.7	Design objects and functions for displaying and handling interactions with various game menus	7	2/7/2020	2/13/2020	100%
3	Develop the main components of the game engine	14	2/14/2020	2/27/2020	100%
3.1	Test and Refine these components	2	2/28/2020	2/29/2020	100%
4	Design basic/sample assets for the game to use	4	3/1/2020	3/4/2020	100%
5	Design player model and animations	6	3/5/2020	3/10/2020	100%
6	Design game world object's models and animations	6	3/11/2020	3/16/2020	100%
7	Test and refine game objects and assets	2	3/17/2020	3/18/2020	100%
8	Design objects and functions for analyzing game music	4	3/19/2020	3/22/2020	100%
9	Design objects and functions to use this data to output proper actions for game objects	6	3/23/2020	3/28/2020	100%
10	Test and refine game objects and assets based on the actions from the game music	4	3/29/2020	4/1/2020	100%
11	*Design additional game objects and functions game mechanics if time is available*				
11.1	Additional player items (weapons, armor, ...) to increase gameplay/player variety	4	4/2/2020	4/5/2020	100%
11.2	Additional game environments and game world objects	5	4/6/2020	4/10/2020	100%
12	Create product demos	14	4/11/2020	4/24/2020	100%



Results and Current State

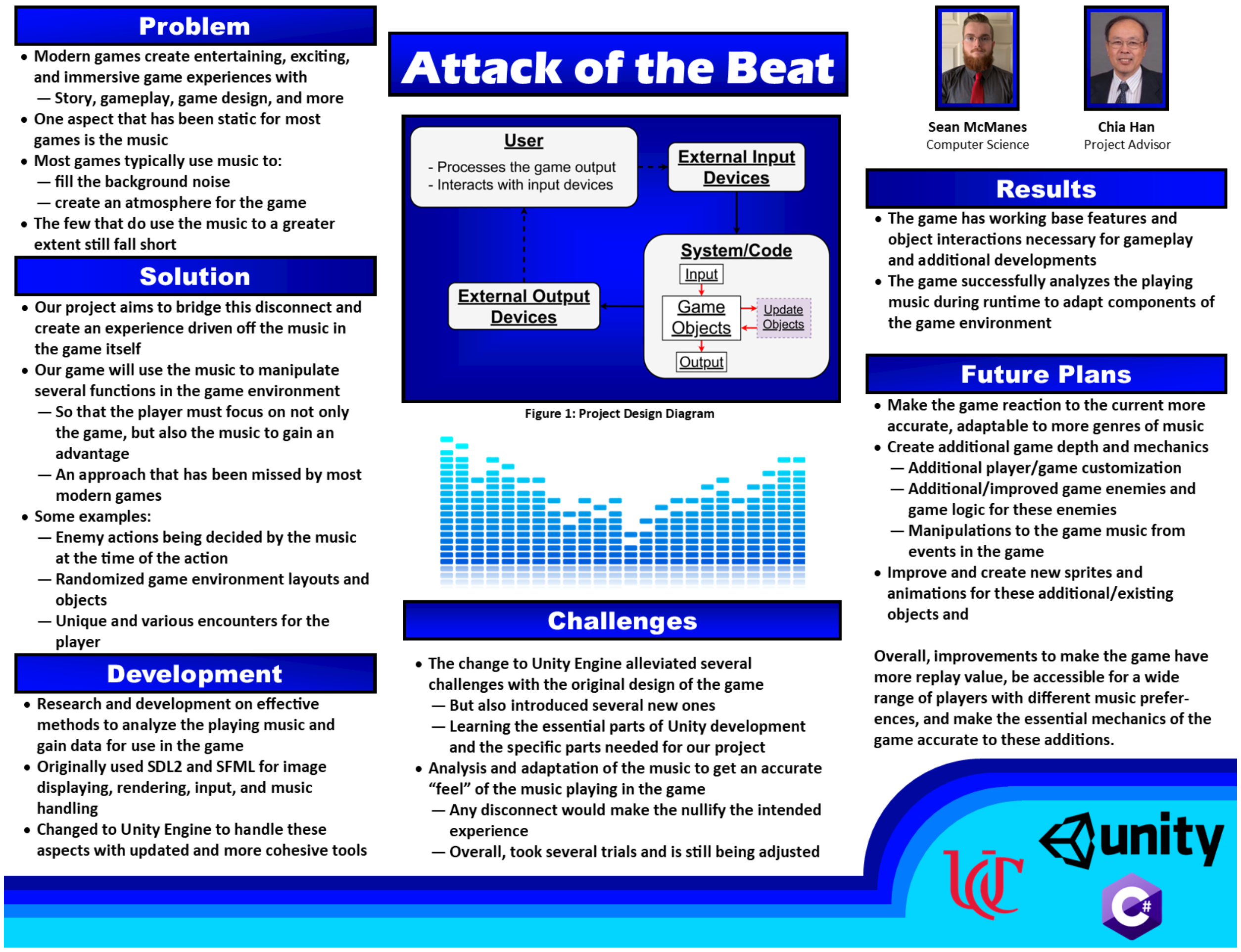
- The game has working base features and object interactions necessary of the gameplay and additional developments
- The game successfully analyzes the playing music during runtime to adapt components of the game environment
- Now that these key components are in their final states, additions and improvements are being made with the time remaining
  - Mainly adding as many game mechanics that were originally desired and getting them to a working state with the game music
  - Improving the music analysis several test music/genres



Challenges

- Project designs were originally based on the use of libraries (such as SDL2 and SFML) to display the game and handle game audio
  - Decided to change to Unity Engine, so developed code had to be converted and adapted to work with it
- The change to Unity Engine made much of the development process easier, but it came with a learning curve since we had never used it
  - Much of the challenge was understanding the general workflow and development in Unity
  - Once the major pieces were created, the remaining parts quickly fell into place
- Analysis and adaptation of the music created the challenge of getting an accurate "feel" of the music in the game
  - Any disconnect would make the game experience seem incorrect
  - This was an aspect that took several trials and is still getting adjusted

Final Expo Poster



Attack of the Beat

Sean McManes
Computer Science

Chia Han
Project Advisor

Problem

- Modern games create entertaining, exciting, and immersive game experiences with
  — Story, gameplay, game design, and more
- One aspect that has been static for most games is the music
- Most games typically use music to:
  — fill the background noise
  — create an atmosphere for the game
- The few that do use the music to a greater extent still fall short

Solution

- Our project aims to bridge this disconnect and create an experience driven off the music in the game itself
- Our game will use the music to manipulate several functions in the game environment
  — So that the player must focus on not only the game, but also the music to gain an advantage
  — An approach that has been missed by most modern games
- Some examples:
  — Enemy actions being decided by the music at the time of the action
  — Randomized game environment layouts and objects
  — Unique and various encounters for the player

Development

- Research and development on effective methods to analyze the playing music and gain data for use in the game
- Originally used SDL2 and SFML for image displaying, rendering, input, and music handling
- Changed to Unity Engine to handle these aspects with updated and more cohesive tools

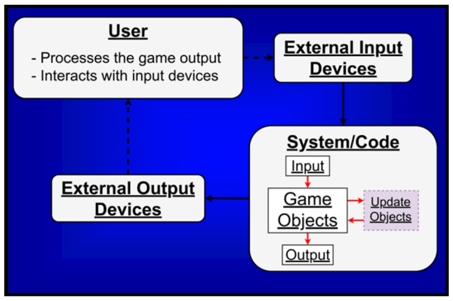


Figure 1: Project Design Diagram

Challenges

- The change to Unity Engine alleviated several challenges with the original design of the game
  — But also introduced several new ones
  — Learning the essential parts of Unity development and the specific parts needed for our project
- Analysis and adaptation of the music to get an accurate "feel" of the music playing in the game
  — Any disconnect would make the nullify the intended experience
  — Overall, took several trials and is still being adjusted

Results

- The game has working base features and object interactions necessary for gameplay and additional developments
- The game successfully analyzes the playing music during runtime to adapt components of the game environment

Future Plans

- Make the game reaction to the current more accurate, adaptable to more genres of music
- Create additional game depth and mechanics
  — Additional player/game customization
  — Additional/improved game enemies and game logic for these enemies
  — Manipulations to the game music from events in the game
- Improve and create new sprites and animations for these additional/existing objects and

Overall, improvements to make the game have more replay value, be accessible for a wide range of players with different music prefer-ences, and make the essential mechanics of the game accurate to these additions.

<u>Self-Assessment Essay: *Fall - Sean McManes*</u>

Attack of the Beat is a project dedicated to creating an entertaining, exciting, and immersive game experience for the player and puts an emphasis on the music in the game. This music will manipulate several functions in the game environment so that the player must focus on not only the game itself but also the game music to gain an advantage. This aspect of gaming has not been approached in most modern games. These games typically use their music to fill the background noise of the game and do almost nothing more than create an atmosphere for the game. The music does not have a direct affect in the game experience. Our project will attempt to bridge this disconnect and create a game experience driven off the music in the game itself.

The experiences that we gained during our academic career will assist our project development immensely. Considering I have not completed any co-op semesters so far, most of my experience comes from the courses that I have completed during my time at the University of Cincinnati and from my own experiences outside of the class. Our project will involve the use of several class types, functions, and established libraries in order to achieve a functioning game. CS 2011 Introduction to Computer Science and CS 2028C Data Structures were the courses that established and expanded my knowledge of programming with C++ and the importance of object designing in a program. This project will require the use of user inputs, contain several game objects that will have to be managed and manipulated, and contain a form of storage for object information and other important data. The skills gained from these courses will especially assist with the overall designing of object types, deciding what data needs to be stored and where, and creating the necessary functions to manipulate this data and interact with other objects.

As mentioned, the project will have to manage various objects in the game environment, store portions of this data for future use, and be able to read stored data as well. This will allow the code of the project itself to stay clean of repetitive data that could be stored elsewhere and read from/written to when needed. My experience from CS 2071 Discrete Structures, CS 4071 Design and Analysis of Algorithms, and CS 4092 Database Design and Development will assist me with these aspects of the project. The objects in the game could vary depending on the scenario. In the worst case, there can be several objects that need to be handled depending on what is happening in the game. In order to do this, the object must be managed in various ways and my experience in these courses will especially help in that design process. I will be able to choose the best ways of storing objects and finding the ones that need to be manipulated based on what the search parameters are and be able to evaluate the code to know if there are perhaps better options that would be more efficient. Outside of these courses, I have experimented with ways of collecting user inputs and being able to display objects and outputs beyond what have been done during my coursework (typically just command line). For this project, the use of libraries such as SDL2 and SFML will be useful for displaying the game and both reading and outputting audio.

Most of my motivation for this project comes from my enjoyment and time spent with video games during my life. The countless experiences that I have gained from them and the communities that they have connected me with have affected several aspects of my academic career path and other aspects of my life. From these experiences, I want to be able to expand upon the aspects that I find the most enjoyable for not only myself, but also for a larger audience. During my time with video games and outside of them, I also gained an appreciation for music. Some games have great soundtracks that add to the atmosphere of the game and can get you truly immersed in the experience. Others seem to fall short or do not fully use the music to add to the gameplay itself. From my experience with both game and music I can see the disconnect between them and this became the foundation for this project.

Our first steps in reaching our goal for this project are to build a working game and game design that will be the backbone for the rest of the project. If the game itself is not interesting to the player or is not functioning properly, then the rest of the project will fall short. Once the basic game framework is set, then the aspects of the game that we want to be manipulated by the music can be decided and implemented. The major aspect we have in mind are enemy attack and movement. In order to manipulate these aspects, the music playing must be analyzed and the data from this interpreted by the game into actions for the objects in the game. These actions will most likely be based on the frequency of the sounds, and their duration, being played, but their effects in game may depend on the objects that we want to be affected. For example, certain enemies may only attack when a range of frequencies are played, and their attack power or types may depend on duration of those frequencies. After these aspects have been refined and are working properly, then we will be able to work on expanding the game's objects further. Some of these ideas will be dependent on the time available for them, but if possible, it would be nice to have more variety in the game world, such as different area types, enemy types, game objects, and so on. Overall, we will know we are done once we have made the experience enjoyable for the player and incorporates our major ideas on what the game music can manipulate to make that experience unique for the player. We could then get feedback from other game enthusiasts to know what aspects of the game succeeded and if there are any areas that could be improved.

Self-Assessment Essay: *Spring - Sean McManes*

The building and completion of all tasks indicated in the project timeline/milestones was completed by me. Unfortunately, my original team member was unable to take the second part of Senior Design this semester, but I will discuss this in more detail later. I applied all the skills that was identified in my initial assessment last Fall, and even gained additional skills that I had not intended at the time. The original plan was to create the game from scratch in C++, as I was most familiar with it, and build a simple engine from it and create the music analysis and game mechanics with this engine and additional libraries. However, due to the additional workload I took in order to get this project finished, I decided to use the Unity Engine to assist with some of the development. This made it possible to get a working base game to expand off with the ideas that I had envisioned. Unity also made the music analysis easily accessible, and in no time, I was able to get the major components developed to drive any additional game objects I wanted to add.

Overall, learning how to create with Unity, since I had no prior experience, was the major obstacle. But once I was past that, trying to get the music analysis to properly mimic the "feel" of the music I tested was the harder obstacle. The solutions that I came up with worked in the scenarios that I have in the game currently, but with other ideas that I had, several things would need to be changed. Specifically, being able to analyze the sound that would be played during the next frame of the game was something that I did not have time to complete, and had to be put on hold so I could finalize other components. If I had more time, this would be the next part that I would develop on, and I believe I could make even more interesting and interactive experiences for the player.

Since I was the only member of this group during this semester, all the group accomplishments were my own and are stated in my assessment above. Last semester, after completing most of the assignments myself and having difficulties getting my partner to participate in a timely manner in said assignments, I knew this semester was going to be a similar story. When I found out that my partner was not taking the course this semester, I was somewhat relieved. It is unfortunate that my experience with groupwork did not go as well as I had hoped, but I have had experiences like this before, and it has never stopped me from completing my goals. And it did not stop me with this project.

Even with this experience, I am far from turned away from group-related projects, and I still view this project as an overall success and highlight of my college career. If I had a team member or two with the same motivation as I do, then I believe this project could have been expanded even further, meeting the future goals I mentioned above and even more functionality that I had envisioned during the planning last semester. Overall, this project showed that motivation from all members is necessary for the whole group to be successful. As every group is only as strong as their weakest link. Once those links either removed or improved, then it is possible for the group to be fully productive and complete its tasks.

Summary of Hours: *Fall*

Date	By	Description	Hours
8/27/2019	Sean	Class Meeting and Project Discussion	1
8/30/2019	Sean	Professional Biography	2
9/3/2019	Sean	Class Meeting and Project Discussion	1
9/6/2019	Sean	Project Description	2
9/8/2019	Sean	Discussion of Overall Game Ideas	1
9/10/2019	Sean	Class Meeting and Project Discussion	1
9/12/2019	Sean	Capstone Assessment	2
9/14/2019	Sean	Researching Game Design Ideas	2
9/17/2019	Sean	Class Meeting and Project Discussion	1
9/18/2019	Sean	Researching and Experimentation with Game Display Design	4
9/18/2019	Sean	User Stories	2
9/20/2019	Sean	Design Diagrams Discussion	2
9/21/2019	Sean	Researching and Experimentation with Object Management	3
9/22/2019	Sean	Design Diagrams Finalizing	2
9/24/2019	Sean	Class Meeting and Project Discussion	1
9/27/2019	Sean	Initial Task List Discussion	2
10/1/2019	Sean	Class Meeting and Project Discussion	1
10/3/2019	Sean	Task List Finalizing	2
10/8/2019	Sean	Class Meeting and Project Discussion	1
10/9/2019	Sean	Timeline	2
10/11/2019	Sean	Effort Matrix	2
10/12/2019	Sean	Continuing the Design of Early Game Display Functions	4
10/15/2019	Sean	Class Meeting and Project Discussion	1
10/18/2019	Sean	Experimenting with Functions to Analyze Game Music	1
10/23/2019	Sean	Slideshow Discussion	2
10/25/2019	Sean	Brainstorming Possible Game Music Manipulations	3
10/30/2019	Sean	Slideshow Finalization and Recording	3
11/6/2019	Sean	Designing High Level Ideas to Implement the Music Manipulations	2
11/12/2019	Sean	Presentation and Class Meeting	2
11/15/2019	Sean	Continuing Game Object Design and Early Testing	2
11/19/2019	Sean	Peer Presentations and Class Meeting/Discussion	2
11/26/2019	Sean	Peer Presentations and Class Meeting/Discussion	2
12/4/2019	Sean	Meeting with Advisor	0.5
		Total Number of Hours	61.5
		Group Bill ($37.50/Hr)	$2306.25

Summary of Hours: *Spring*

Date	By	Description	Hours
12/16/2019	Sean	Test Assets Development	5
12/20/2019	Sean	Map Creater and Initial Asset Creation	5
12/30/2019	Sean	Initial Audio Analysis Function Creation	5
1/2/2020	Sean	SFML and SDL Game Display Function Updating	5
1/15/2020	Sean	Class Meeting and Planning	1
1/19/2020	Sean	Test Plans	3
1/24/2020	Sean	User Documentation	3
2/5/2020	Sean	Class Meeting and Planning	1
2/10/2020	Sean	Game Object and Music Initial Combination Experimentation	5
2/14/2020	Sean	PowerPoint Documentation	3
2/20/2020	Sean	Unity Research and Initial Conversion	5
2/26/2020	Sean	Expo Poster Draft	3
3/1/2020	Sean	Unity Initial Base Game Setup and Experimentation	5
3/4/2020	Sean	Class Meeting and Reviewing	1
3/10/2020	Sean	Unity Map Creator Conversion	3
3/16/2020	Sean	Unity Player Asset Creation and Adjustments	4
3/18/2020	Sean	Unity Player Animation Development	3
3/20/2020	Sean	Unity Player Movement and Action Development	5
3/21/2020	Sean	Unity Sound Analysis Development	3
3/22/2020	Sean	Unity Visual Sound Analyzer Development	5
3/23/2020	Sean	Poster Final Draft	2
3/23/2020	Sean	Unity Music-Manipulated Function Development	5
3/25/2020	Sean	Unity Projectile Attempt	5
3/26/2020	Sean	Unity Data Prediction Experimentation	3
3/30/2020	Sean	Unity Laser Object Development	5
4/5/2020	Sean	Unity Laser Object Improvement	3
4/10/2020	Sean	Final Self-Assessment	2
4/16/2020	Sean	Expo Demo Video	2
4/18/2020	Sean	Final Report	5
4/20/2020	Sean	Final Revisions and Improvement	5
		Total Number of Hours	110
		Group Bill ($37.50/Hr)	$4125.00

Appendix

All project files, references, and project updates can be found on the Attack of the Beat Github Repository located at the following link.

https://github.com/WrathOfRa/AotB