



INSTRUMENT (GAME) ESSAY

ADAD7005 – Development – Element 2B

Matthew Buckland
Student No: 2010910

Table of Contents

Title	Pg. No
<u>1. Approach Taken to Create the Game</u>	2
<u>2. Principles of How the Game Work</u>	4
2.1 Notes	4
2.2 Truck	4
2.3 Background Music and “You Win”	4
2.4 Road (Stave) and Scenery	5
<u>3. Issues Encountered</u>	6
<u>4. Reflections</u>	6
<u>5. Improvements</u>	6
5.1 More Levels	6
5.2 Fix Script for Truck	7
5.3 Cosmetic Improvements	7
5.4 Easy, Medium, and Difficult modes	8
5.5 Builds for iOS and Android devices (and an easily accessible Mac version)	8
5.5.1 Building for Mobile Devices	8
5.5.2 Building for Mac	8
5.6 Supplemental Group Instruction	9
5.7 Audio	9
<u>6. Existing Commercial Games</u>	10
6.1 TuneTrain (Bravura « Everyone can create music!, 2021)	10
6.2 Musical Me! (Musical Me! , 2021)	10
6.3 Piano Dust Buster 2 (When The Saints - 2 Hands, 2021)	11
6.4 Other Games	11
<u>7. User feedback</u>	12
<u>8. Conclusion</u>	12
<u>9. Bibliography</u>	14
<u>10. Appendices</u>	16

1. Approach Taken to Create the Game

I decided to make a serious game. Wikipedia defines a serious game as “A serious game or applied game is a game designed for a primary purpose other than pure entertainment” (Serious game - Wikipedia, 2021) I used the game development tool Unity, and as I have been a gamer for most of my life, game design is something I have always wanted to try. Me being on a Music degree, I decided to base my game around Music. The basic premise of the game is to teach primary school aged children the basics of music theory. The game’s objective is to hit all the notes along the stave, resulting in a “You Win” condition. I created a Project Design Document (PDD) for the game, which can be found in Appendix A. The working title for the game is currently Music Kart.

When developing this idea, I thought at the time it was a relatively simple idea to execute. However, I was soon to find out that this was not the case and threw myself into learning how to script. I have been using the Unity Learn course as an introduction to programming, and to date I am currently halfway through the Junior Programme course (from which I can obtain certification which is recognised within the industry). The course also teaches the student how to navigate within Unity.

I have used FMOD Studio (FMOD) for the sound design. My main source for learning this software is “Game Audio with FMOD and Unity” by Ciaran Robinson (Ciaran Robinson, 2019), and of course, Google search. The piano key sounds, background music (The Wheels on the Bus) and “Ta-Da” sound within the game were created by me in Ableton. The engine sounds I obtained as a free asset pack for completing a unit of the Unity Learn programme. The engine sound is three different (but close) engines to simulate an engine accelerating. This effect was achieved using the “Speed Parameter” in FMOD, as well as using pitch modulation.

The game is built from a mixture of prebuilt and my own assets:

- The truck was used in the first Unity tutorial to learn how to code movement. I have had to rewrite this code, as it was using the Transform.Translate method. This meant that the FMOD speed parameter would not work, as Transform.Translate does not use Unity’s physics engine.
- The notes I found for free online by doing a Google search
- The roads were found on the Unity Asset Store
- The stave image was imported and applied to the road
- The scenery around the track (Piano keyboard, green “grass” areas) were made by me in Unity using 3D objects.

Research on coding was done via the Unity Learn online Junior Programmer course (Unity Learn, n.d.), and the Unity Scripting Forum (Unity Forum, n.d.)

I also wanted to create a music game with a low barrier of entry (i.e., for young children to learn music theory). The article *TuneTrain: Incidental Learning Through Interactive Music Creation* (Lee et al., 2013), written for the 2013 International IEEE Games Conference, defines two categories of music creation applications – Tools and Toys. The Tools category contains programmes such as Ableton, Logic and Pro Tools, and these have a high barrier to entry. An example of a Toy application would be Jam for iOS, in which the user chooses musical style and autotune parameters. The application then adds accompaniment automatically to create a song.

My idea for Music Kart, is that young children (from ages 4+) would be able to explore and create music, through game, by combining the Tools and the Toys. I want it to inspire creativity and empower children to create music, whilst giving them some of the basic principles of music theory.

2. Principles of How the Game Works

The objective of Music Kart is to hit all the notes along the stave to achieve the “You Win” condition. The notes are arranged along the stave (road) in C Major Scale ascending and descending.

2.1 Notes

When each note is hit it will emit the sound of the relative piano key. This is achieved through using the DestroyObject method; once the object is destroyed the script tells FMOD to PlayOneShot the relative piano key. The individual piano key sounds were made in Ableton using the Spitfire Audio LABS Soft Piano free VST. These were then bounced and imported to FMOD. The notes use Rigidbody and Box Collider so that they can interact with other Rigidbody and Box Collider objects (such as the truck).

2.2 Truck

Movement of the truck is handled using the Forward and Horizontal Input methods. Arrow Keys Up and Down move the truck forwards and backwards, and Arrow Keys Left and Right turn the truck. As mentioned above, originally this used the Transform.Translate method, however FMOD Studio parameters would not work with this, thus the engine would not alter pitch when acceleration is applied. I rewrote the script (discussed in [5. Improvements](#)). The truck uses Rigidbody and Collider so that it can interact with other objects within the game and allows the scripts to work.

The sound of the engine alters pitch when acceleration is applied using the “Arrow Up” key is pressed down. It decelerates when the “Arrow Up” key is released. This was achieved by setting the “Speed Parameter” in FMOD to match the max speed the truck can go in Unity (this is 15f). (See Engine Audio Script and Car Controller Controller Script in Appendices B and C)

2.3 Background Music and “You Win”

The background music was made by me in Ableton using Serum, drum samples and the Spitfire Audio BBC Orchestra free VST. I applied some Redux to the drum buss in Ableton to give it that old school game sound. I then bounced the track out and imported into FMOD.

The background music (The Wheels on the Bus) I thought was quite fitting and adds some joviality to the game. The music starts when the game starts. This is achieved through the Background Music script (See Appendix D). An instance of FMOD is created on Game Start and plays on a loop until the game objective is met. I have implemented some sidechain compression in FMOD so that when the notes are hit, the background music ducks so that the piano sound can be heard. I have also applied some reverb in FMOD. The music stops when the condition of all the notes being destroyed has been met. This is achieved through the script counting how many Note Scripts there are in the game. If the Note Script = Null, then the background music stops, the “Ta-Da” one shot is played, and the text “You Win!” appears on screen.

2.4 *Road (Stave) and Scenery*

These items all use Box Colliders to ensure that the truck does not fall through/off the road. I found an image of a stave on Google and imported the image to Unity. This was then applied to the road using Unity's Textures and Materials functions. The background scenery (mountains, trees, and mushrooms) was found on the Unity Asset store.

Short Level

I decided on making the level short in duration, track length and the number of objectives to complete. From reading the subject regarding attention spans for children, I have found that children at age 3 typically can focus their attention for around 15 to 20 mins. (Sochocka, Bednarska and Grabska-Gradzińska, 2016, p 2). This means for future levels of the game; short durations will apply.

3. Issues Encountered

One of the major issues I had whilst developing this game was that I had my truck originally using the transform.Translate method to move. This does not use Unity's physics engine, meaning that the SFX I had created in FMOD would not work. The FMOD speed parameter uses the physics engine in Unity to tell it how fast along the speed parameter it needs to go, how long, and when to return to the "Idle" setting.

Through research, I came across a tutorial that would enable to move my truck using the physics engine without too much coding. The basic premise is that the truck is placed on a sphere with a Rigidbody, and the Engine Audio script is attached to the sphere so that it is now moving through Unity physically. This enabled me to get the audio working, and it also gives the movement of the truck an arcade like quality.

4. Reflections

The game as it stands in its current form is what I had envisioned (See PDD in Appendix A), and on a personal level I am incredibly pleased with the result. It needs a lot more work, and there are lots of improvements that I want to make to the game (see **5. Improvements** below).

On reflection and from researching games that are aimed at the same target market, I have determined that maybe a side scrolling 2D game will work better (especially for iOS and Android devices). I would develop the 2D version for mobile devices, and the full 3D version for PC/Mac.

5. Improvements

Below are improvements that I would like to make to Music Kart and will be looking at doing these as a personal project for my own development. I am planning to work on this after I have completed my master's degree.

5.1 More Levels

The current game has 1 level. Here is a breakdown of the level structure that I have envisioned for my game:

Level 1 – Scales C Major Scale Ascending/Descending

Level 2 – Scales C Minor Scale Ascending/Descending

Level 3 – Rhythm – I am currently exploring the option to make the truck jump, so it would jump when the user inputs the correct key, and the aim is to time it with a beat.

Level 4 – Melody – User must drive the car to create a melody.

Level 5 – A level in which the user can create their first song, which can be downloaded as an mp3.

5.2 Fix Script for Truck

Currently the truck uses a sphere to move, and whilst this gives it an arcade type of feel, it may be too wild for the target audience. I would either completely re-write the script for the truck's movement or dial down the parameters on the current script. I also need to fix the issue whereby once the game loads, the sphere drops underneath the truck, making the truck float above the track. Whilst I could get around this by replacing the truck with a hovercraft, I would prefer the truck. I would also like to get the wheels spinning and to turn with the truck. I have a script ready for this, however I need to work a few things out on it, and so was not able to implement in time for this project.

The feedback I received from testing (see Feedback below) was that the controls were too difficult for children to use. I have amended the control system so that acceleration is now performed with the Space key rather than the Up Arrow key. I have also implemented the ability to turn without having to accelerate, which was another issue brought up from testing.

5.3 Cosmetic Improvements

I have used mainly pre-made assets for this game, and I would like to hire an art designer (or maybe learn myself) to create original assets. This would give the game a stronger brand identity.

I would also want to create a script whereby the keys on the piano light up once the note has been hit. I feel this would work better in a side scrolling environment, however I feel the truck is not moving too quickly in the current version so that the user would miss the key being highlighted. In conjunction with this, I would also want to make the name of the note that is hit float up (so when the C note is hit, a C will appear above the car, float up and fade out. This was highlighted in feedback from testing carried out by the target audience (and their parents, see **7. User Feedback** below). I think some particle effects would also work quite well for when the notes disappear.

In terms of scenery, I have an idea for it to take inspiration from Nintendo's racing game Mario Kart, but using colour palettes and objects suitable for the target audience. Another idea would be to use popular fairy tales as scenery, as it would help children to focus on the screen. (Sochocka, Bednarska and Grabska-Gradzińska, 2016, p 3)

5.4 Easy, Medium, and Difficult modes

I would like to implement different modes of difficulty. One of the parents of the children that were involved in the test recommended that the notes be hit in sequential order. If the notes are hit this way, the level is completed. If not, the level restarts, and the user can try again. This would be the basic rule throughout the game, that there is a condition to be met. Then I would implement the following levels of difficulty:

- Easy – All notes shown
- Medium – Some notes missing
- Hard – All notes missing

This would need more consideration, particularly with the other types of level mentioned above (e.g., Rhythm and Melody), but I think it is a good starting point to work on.

5.5 Builds for iOS and Android devices (and an easily accessible Mac version)

“The iPad is an ideal platform for children to use because touch input is very natural to them”. (Lee et al., 2013, p 1). One of the feedback items from testing was that the target market mainly uses smart devices (iPhones and tablets) and not Windows or Mac computers. I would aim to make a build of this game for iOS and Android. This would be the side scrolling 2D version.

5.5.1 Building for Mobile Devices

Building for Android would involve including scripts such as the `Permission.HasUserAuthorizedPermission` function (Unity Technologies, 2021), and the Native (C++) plug-ins for Android using Android NDK (Unity Technologies, 2021).

Building for iOS would involve setting up an Apple Developer account. It would also include building an XCode Project. This project will compile and sign the app before being deployed on the iOS device (Unity Technologies, 2021). The game project files will also need amending (e.g., Classes folder and Data folder).

I would also need to alter the way in which user input is handled in the current build. To build the game so that users can use the touch screen on their devices, I would need to implement the `Input.Touches` array (Unity Technologies, 2021)

5.5.2 Building for Mac

From researching for Mac builds, all apps running on macOS 10.15 Catalina are required to be notarized. (How to notarize a Unity build for MacOS 10.15 Catalina, 2021). This would be done post build within the XCode Project, using XCode command line tools.

5.6 Supplemental Group Instruction

Another improvement I would like to add is the idea of being able to use the game in a school setting. The game would come with a Supplemental Group Instruction pack, which would be used by teachers. The idea is that the students would play a level, and then a group discussion would follow to further explore the music concepts discovered in game. However, for the users that are pre-school, the learning would probably be more subliminal (i.e., they are hitting objects that make a sound that ascends/descends along a black and white road, and something lights up at the same time).

5.7 Audio

One last improvement for me to make is the audio. I am not entirely happy with the engine audio, as there is a delay from when the car stops to the engine descending in pitch. I will need to use FMOD more so that I can fully understand how the parameter function works and how to dial in the controls to get the results desired. Also, the mix needs some work. I tested the game with headphones, and the piano notes could be clearly heard through the mix when hit. However, the users that tested the game did not use headphones. I will need to work on the mix so that it is audible through laptop speakers (and iOS/Android devices).

6. Existing Commercial Games

6.1 TuneTrain (Bravura « Everyone can create music!, 2021)

TuneTrain is a game developed by Bravura for iOS and Android. Its target market is for children 8 years old and up. The objective of the game is to connect a series of people (representing musical notes) that are at different heights (representing pitch), and the distance between (representing note lengths). The line that connects the people (the Train) represents a musical melody. I viewed footage of the game being played on YouTube (Appar, 2013)

What I think is particularly clever about TuneTrain is that it uses a method developed by Jazz bassist Ron Carter. He “developed a method of teaching improvisation by presenting chord structures to students and allowing them to draw lines connecting parts of the chords together” (Lee et al., 2013, p 2) TuneTrain uses this method in that users connect to the next note via drawing a line.

One other feature, that I would like to add to my game is to simulate the ascending and descending scales. TuneTrain uses building heights to show different pitches. For Music Kart, the track would rise a hill for the ascending scale and come back down for the descending scale.

TuneTrain not designed as a game. There is no level progression or win condition, however its focus is that of allowing the user to create music whilst incidentally learning.

The game also comes with a teacher’s guide, so that it can be used in the school environment.

6.2 Musical Me! (Musical Me! |, 2021)

Musical Me, developed by DuckDuck Moose is a suite of mini games exploring the following: Rhythm, Memory, Dance, Instruments (play along), and Notes (create your own music). From watching game footage on YouTube (GamesClubFun, 2017), this game appears to be the closest out of the games I have researched that resembles what I would like my finished game to be. The animations are childlike in nature, and the game is narrated by a child. Each level explores a different musical concept.

The sound design is very relaxed, but loud and clear. The game includes 14 popular children’s songs. This is something I would like to use in my game. Currently the first level is “The Wheels on the Bus”, and each level would have a different children’s song as background music or as part of the main objective of the level.

Musical Me! is the only game I found that was reviewed on commonsense.org, a non-profit organization that provides trustworthy information on learning apps for parents and teachers. It has an overall rating of 3 stars, which is broken down into Engagement, Pedagogy, and Support (Musical Me - by Duck Duck Moose Review for Teachers | Common Sense Education, 2021)

6.3 Piano Dust Buster 2 (*When The Saints - 2 Hands*, 2021)

Piano Dust Buster, developed by JoyTunes, is a game based around the piano, in which users must help granny dust off her piano. They do this by playing the correct notes on the keyboard as they appear on screen. This game works on iOS as a touch screen game but can also be used with a real piano keyboard.

From watching game footage online, this game uses more contemporary music compared with the two games above. One such example is Clocks by Coldplay (Magic Productions, 2013).

The aesthetic of this game is that of a more traditional music teaching environment. It includes two modes – Germ Attack (Easy) and Staff Master (Advanced). It also includes a point scoring system, and a groove meter, making it the most game like out of the games I have researched.

6.4 Other Games

Whilst conducting research for this essay, I found that a lot of serious music games aimed at this market were either Flash games (which is no longer supported by Adobe since 31 December 2020) (www.adobe.com, n.d.), or were no longer available on the App Store.

7. User feedback

Ryleigh (Aged 6) – Feedback via parents

Feedback	Response	Implemented Changes (Y/N) Comments
Controls too difficult to use – arrow keys	Changed forward input to Space bar	Y Latest Build
Cannot hear piano keys over background music	Lowered background music level in FMOD	Y Latest Build
Name of note could appear on screen when hit	See Improvements section above	N Research Paper
Get notes in right order, otherwise lap resets	See Improvements section above	N Research Paper
Difficulty – as player progresses, notes disappear etc.	See Improvements section above	N Research Paper

Florence (Aged 4) – Feedback via parents

Feedback	Response	Implemented Changes (Y/N) Comments
Understood objective	Amended target audience from 5+ to 4+ (more testing needed on a wider range)	Y Research Paper
Controls too difficult to use – arrow keys and not being able to turn on the spot	Changed forward input to Space bar in latest build. Amended script so that the turn function is now available without having to accelerate.	Y Latest Build
Cannot hear piano keys over background music	Lowered background music level in FMOD	Y Latest Build
Managed to break game – fell down gap in scenery	Unable to replicate in testing	N More testing required

8. Conclusion

In conclusion, I have built a working prototype of a game using the Unity Game Engine and FMOD Studio. The game in its current form is what I intended to design and build. The games objectives are for the player to hit all the notes along a stave, using a truck. I have encountered many trials and tribulations along the way (mostly to do with coding), however I am extremely pleased with the result.

I would like to add more levels to the game exploring the different elements of music theory, which I think will be an interesting challenge considering the target market, platform and ensuring that the content is both teaching and allowing the user to create.

From my research and user testing, there is a market for this kind of application. I will consider two versions of this game, a 3D version for PC/Mac, and a side scrolling 2D version for iOS and Android.

The testing of the game by the users who were of target market age was extremely valuable and highlighted that it is necessary to put oneself in the target players shoes. If the target market cannot use the controls to play the game, they will not play the game. This is something for me to consider more in the early design stages of the project.

9. Bibliography

- www.adobe.com. (n.d.). Adobe Flash Player End of Life. [online] Available at: <https://www.adobe.com/uk/products/flashplayer/end-of-life.html> [Accessed 23 Mar. 2021].
- Appar, P., 2013. TuneTrain - Skapa melodier. [video] Available at: <https://www.youtube.com/watch?v=nNf03m11wM0> [Accessed 23 March 2021].
- Ciaran Robinson (2019). Game audio with FMOD and Unity. New York, Ny: Routledge.
- Common Sense Education. 2021. Musical Me - by Duck Duck Moose Review for Teachers | Common Sense Education. [online] Available at: <https://www.commonsense.org/education/app/musical-me-by-duck-duck-moose> [Accessed 23 March 2021].
- Duckduckmoose.com. 2021. Musical Me! | . [online] Available at: <https://www.duckduckmoose.com/educational-iphone-itouch-apps-for-kids/musical-me/> [Accessed 23 March 2021].
- En.wikipedia.org. 2021. *Serious game* - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Serious_game [Accessed 25 March 2021].
- Etc.cmu.edu. 2021. Bravura « Everyone can create music!. [online] Available at: <https://www.etc.cmu.edu/projects/bravura/> [Accessed 23 March 2021].
- GamesClubFun, 2017. MUSICAL ME! - BY DUCK DUCK MOOSE KIDS GAME APP FOR AGES 2-6 - MUSICAL WORLD WITH MOZARELLA MOUSE. [video] Available at: <https://www.youtube.com/watch?v=XoBwGAH4HaM> [Accessed 23 March 2021].
- Gist. 2021. How to notarize a Unity build for MacOS 10.15 Catalina. [online] Available at: <https://gist.github.com/dpid/270bdb6c1011fe07211edf431b2d0fe4> [Accessed 23 March 2021].
- Lee, J., Gea, A., Yang, C., Deciron, R., Lee, M., Lu, P. and Xie, M., 2013. TuneTrain: Incidental Learning Through Interactive Music Creation. [eBook] Redwood City: Carnegie Mellon University, pp.1-4. Available at: <https://ieeexplore.ieee.org/document/6659139> [Accessed 23 March 2021].
- Magic Productions, 2013. Coldplay - Clocks - Piano Dust Buster 2 Tutorial. [video] Available at: <https://www.youtube.com/watch?v=BMbB74II5eI> [Accessed 23 March 2021].
- Sochocka, A., Bednarska, M. and Grabska-Gradzińska, I., 2016. Importance of computer games in children's life in the 21st century. Bio-Algorithms and Med-Systems, 12(4).
- Technologies, U., 2021. Unity - Scripting API: Android.Permission.HasUserAuthorizedPermission. [online] Docs.unity3d.com. Available at: <https://docs.unity3d.com/ScriptReference/Android.Permission.HasUserAuthorizedPermission.html> [Accessed 23 March 2021].
- Technologies, U., 2021. Unity - Manual: Native (C++) plug-ins for Android. [online] Docs.unity3d.com. Available at: <https://docs.unity3d.com/Manual/AndroidNativePlugins.html> [Accessed 23 March 2021].
- Technologies, U., 2021. Unity - Manual: Getting started with iOS development. [online] Docs.unity3d.com. Available at: <https://docs.unity3d.com/Manual/iphone-GettingStarted.html> [Accessed 23 March 2021].

Technologies, U. (n.d.). Unity - Manual: Mobile device input. [online] docs.unity3d.com. Available at: <https://docs.unity3d.com/Manual/MobileInput.html>.

Unity Learn. (n.d.). Junior Programmer. [online] Available at: <https://learn.unity.com/pathway/junior-programmer> [Accessed 23 Mar. 2021].

Unity Forum. (n.d.). Scripting. [online] Available at: <https://forum.unity.com/forums/scripting.12/> [Accessed 23 Mar. 2021].

10. Appendices

Title	Pg. No.
Appendix A – Project Design Document	17
Appendix B – Engine Audio Script	20
Appendix C – Car Controller Script	21
Appendix D – Background Music Script	22

Project Design Document

13/02/2021

Matt Buckland

Project Concept

1

Player Control

You control a

Car

in this

isometric

game

where

Arrow keys

makes the player

Turn in all directions

2

Basic Gameplay

During the game,

Notes

appear

from

Floor of the screen

and the goal of the game is to

Is to hit all the notes

3

Sound & Effects

There will be sound effects

when the notes are hit

and particle effects

when the notes are hit

[optional] There will also be

4 Gameplay Mechanics	As the game progresses,		making it
	Each level will explore a different part of basic music theory		A learning game
	[optional] There will also be description of any other gameplay mechanic(s) and their effect on the game.		

5 User Interface	The	will	whenever
	level	Complete	All of the notes are hit
	At the start of the game, the title Music Kart		

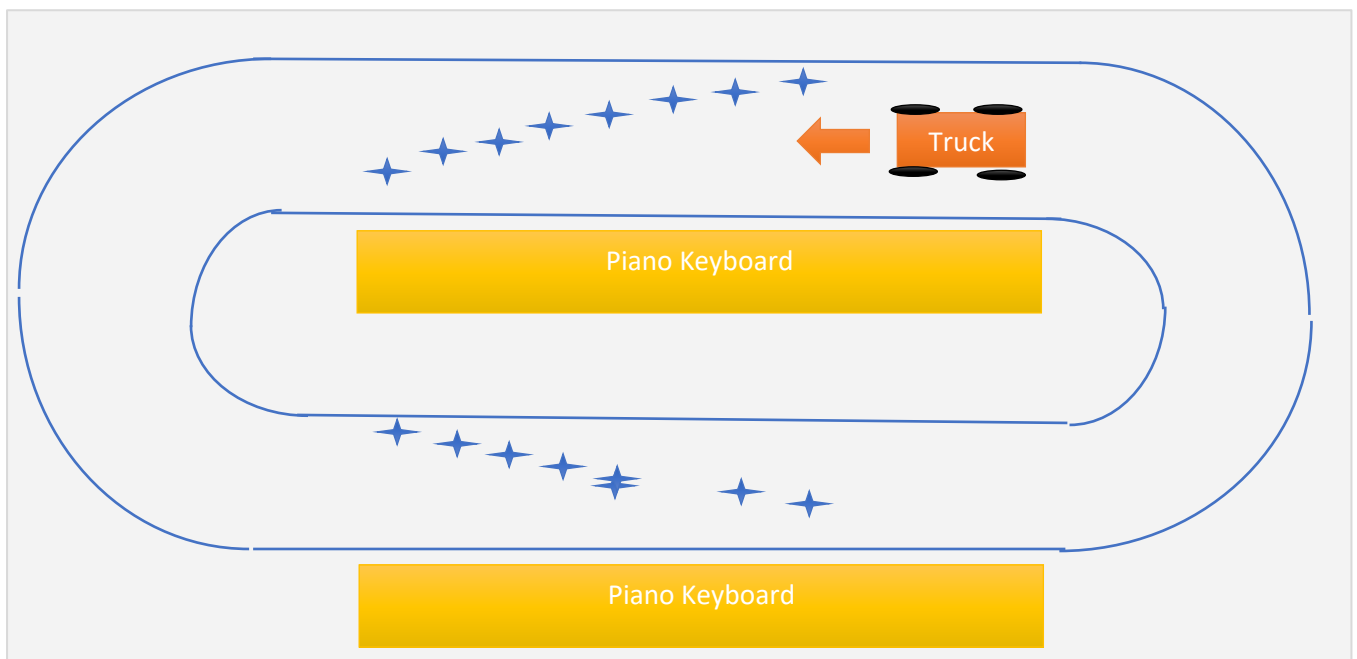
6 Other Features	Aimed at Primary School aged students to teach the basics of music theory		
	1st level explores C Maj Scale ascending/descending		
	2nd level explores C Min scale ascending/descending		
	3rd Level explores rhythm		
	4th combines the two to allow the student to make their first song		

Project Timeline

Milestone	Description	Due
#1	- Player controls/camera and circular track added	02/28

#2	- <i>Audio in FMOD integrated</i>	03/07
#3	- <i>Notes, scenery, and game play objectives added</i>	03/07
#4	<i>Beta test of prototype</i>	03/21
#5	- <i>Working prototype</i>	03/26
Backlog	- <i>2-4th Levels - not a part of the prototype</i> - <i>Complete visual assets- not a part of the prototype</i> - <i>UI Menus - not a part of the prototype</i>	12/31

Project Sketch



Appendix B – Engine Audio Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EngineAudioScript : MonoBehaviour
{
    private float truckSpeed;
    private FMOD.Studio.EventInstance truckEngine;
    private FMOD.Studio.ParameterInstance vehicleSpeed;

    // Start is called before the first frame update
    void Start()
    {
        truckEngine =
FMODUnity.RuntimeManager.CreateInstance("event:/Engine/TruckEngine");
        truckEngine.start();
        truckEngine.getParameter("VehicleSpeed", out vehicleSpeed);
    }

    // Update is called once per frame
    void Update()
    {
        truckSpeed = gameObject.GetComponent<Rigidbody>().velocity.magnitude;
        vehicleSpeed.setValue(truckSpeed);
    }
}
```

Appendix C – Car Controller Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CarControllerScript : MonoBehaviour
{
    public Rigidbody theRB;

    public float forwardAccel = 8f, reverseAccel = 4f, maxSpeed = 15f, turnStrength =
50, gravityForce = 10f;
    public int goal = 16;
    public int destroyed = 0;

    private float speedInput, turnInput;

    // Start is called before the first frame update
    void Start()
    {
        theRB.transform.parent = null;
    }

    // Update is called once per frame
    void Update()
    {
        speedInput = 0f;
        if (Input.GetAxis("Vertical") > 0)
        {
            speedInput = Input.GetAxis("Vertical") * forwardAccel * 500f;
        }
        else if (Input.GetAxis("Vertical") < 0)
        {
            speedInput = Input.GetAxis("Vertical") * reverseAccel * 500f;
        }

        turnInput = Input.GetAxis("Horizontal");
        transform.rotation = Quaternion.Euler(transform.rotation.eulerAngles + new
Vector3(0f, turnInput * turnStrength * Time.deltaTime , 0f));

        transform.position = theRB.transform.position;

    }
    private void FixedUpdate()
    {
        if (Mathf.Abs(speedInput) > 0)
        {
            theRB.AddForce(transform.forward * speedInput);
        }
    }
    public void NoteDestroyed()
    {
        destroyed++;
        if (destroyed >= goal)
        {
            theRB.velocity = Vector3.zero;
        }
    }
}
```

Appendix D – Background Music Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BackgroundMusicScript : MonoBehaviour
{
    public FMOD.Studio.EventInstance backgroundMusic;

    NoteGoal ng;
    public int goal;
    private bool runOnceCheck = false;

    // Start is called before the first frame update
    void Start()
    {
        backgroundMusic =
FMODUnity.RuntimeManager.CreateInstance("event:/Background_Music/Background_Music");
        backgroundMusic.start();
        ng = GetComponent<NoteGoal>();
    }

    // Update is called once per frame
    void Update()
    {
        if (GameObject.FindObjectOfType<NoteScript>() == null)
        {
            if (runOnceCheck == false)
            {
                backgroundMusic.stop(FMOD.Studio.STOP_MODE.ALLOWFADEOUT);
                FMODUnity.RuntimeManager.PlayOneShot("event:/Victory/Victory");
                runOnceCheck = true;
            }
        }
    }
}
```