Visual Wave Project

Report

Team: Class 2 - Team 1

Members:

- 20182705 고주형
- 20185784 김호성
- 20182610 손희승
- 20162874 이준협
- 20142611 이하람

Brief Project Description:

This program is MP3 Player that has music visualizer and user can play simple game in that visualized music.

So, our program has simple MP3 functions (play, stop, pause, playlist, ···) and special functions which are visualizing music and with that visualized graphic that user can play game.

We used Unity with WinAPI for our project.

How to play

[Play Music and Press Enter] Visualizer starts and in that map player is spawned.

To move you can press w, a, s, d. To rotate you can press q, e. To jump you can press spacebar.

If you catch light, score increases.

.

How to compile and execute:

We used Unity (2019.2.9f1) so, it should be built within Unity (2019.2.9f1).

In Unity there is Build Menu. In this menu, it provides all kinds of Platform build but, our project uses WinAPI so only Window build is available.

System requirement for compilation and execution:

- Target OS: Windows.
  It uses WinAPI so Operating System should be Windows.
- Unity Version: 2019.2.9f1
  To avoid unity version conflict, you should build our project file with above Unity Version.

Description on functionality of your software:

1. Class Controller: It is responsible for logic of Controller which start, stop, and skips music. And also, it is responsible for UI of that.

   A. void OnStartButtonClicked()
   : When StartButton is pressed, depending on music playing state, it calls MusicStart or MusicStop function.

   B. void MusicStart()
   : It starts music. Music is started at the point of time which is saved on Playlist.

   C. void MusicStop()
   : Stops music and saves the point of time in Playlist.

   D. void MusicSkipToNext()
   : It calls curPlaylist's SkipToNext() function. Depending on bool value it Starts or Stops music. (* curPlaylist is current Playlist)

      i. True: It means we can play next music, so it calls MusicStart function. (Playing next or previous music is handled inside of the Playlist)

      ii. False: It means we cannot play next music, so, it is end of the Playlist. In this case, it calls MusicStop and then calls Init function.

   E. void MusicSkipToPrev()
   : It calls curPlaylist's SkipToPrev function.

      i. True: It means we can play previous music, so it calls MusicStart function.

      ii. False: It means we cannot play previous music, so, it is at the first of the Playlist. In this case, it also calls MusicStart.

2. Class Playlist: It manages Music.

   A. void Init()
      : It initializes Playlist. (Variables such as, current music index, playtime are initiated into zero)

   B. bool SkipToNext()
      : If curMusicIndex is not more or equals then maxCount, plus 1. And then assign 0 to playtime.

   C. bool SkipToPrev()
      : If curMusicIndex is not less then minCount, subtract 1. And then assign 0 to playtime.

   D. void AddMusic(string)
      : It receives filePath and instatiates Music instance and save it into Playlist.

   E. void DeleteMusic(int)
      : It receives index and deletes Music by that index.

   F. string ToString()
      : It is called to save Playlist's information. It return string into specific format that we made.

   G. LoadPlaylist(string)
      : It receives formatted string which is converted data from Playlist's data. By the format we've chose it initiates Playlist.

3. Class DataManager: It manages Playlist. It has add, delete, save, load, ··· functions.

   A. void AddPlaylist(Playlist)
      : It receives Playlist and save it. It is called when Playlist is instantiated.

   B. void DeletePlaylist(int)
      : It receives index and according to that index it deletes Playlist.

   C. void SavePlaylist()
      : It is called when inner data is added, deleted, or any kinds of modifiedification is occurred. It saves data into specific format string.

   D. void LoadPlaylists()
      : It is called when Application is started. It loads all kinds of save data.

4. Class UIManager: It manages UI that is Poped Up when PlaylistButton is pressed. In PopUp UI, it has Add or Delete Playlist, Add or Delete or Select Music functions.
   Also, It uses enum DataType.
   (```enum DataType {NONE, PLAYLIST, MUSIC}```)

   A. void LoadButtons(DataType)
      : It loads corresponding DataType's Button.

   B. void OnButtonClicked(int)
      : It receives corresponding button's index. By that index if the button is clicked twice in 0.25 second, it calls OnButtonDoubleClicked function.

   C. void OnButtonDoubleClicked()
      : It proceeds differently by curDataType which is DataType that current UIManager is indicating.

      i. DataType.PLAYLIST: It saves selected Playlist and with DataType.MUSIC argument it calls LoadButtons function.

      ii. DataType.MUSIC: It sends selected music's index to Controller, and in corresponding Playlist, it plays music from the given index's music.

5. Class BPMManager: It is responsible for beat related thing which are calculating current song's BPM, updating full beat, updating half-quad beat.

   A. BeatDetection()
      : It detects full beat using bpm and half-quad beat dividing full beat interval into 8 again. (This half-quad beat is used for visualizing)

   B. UpdateBPM()
      : It calculates current music clip's BPM using UniBpmAnalyzer(we used OpenSource analyzer for beat sophisticated BPM detection: https://github.com/WestHillApps/UniBpmAnalyzer) and updates current BPM state.

6. Class Visualize Color: It is responsible for changing color base on beat.

   A. Paint()
      : Make the painting color ratio to 1 so we can see the painted 3D objects on beat. Painted objects slowly get its original color.

   B. CheckBeat():
      It check beat so that we can paint color depending on beat.

7. Class Visualize Size: It is responsible for changing size base on beat.

   A. SizeUp()
      : Make the size to set grow value so we can see the increased 3D objects on beat. Sized up objects slowly get its original size.

   B. CheckBeat():
      It check beat so that we can paint color depending on beat.

8. Class Visualizer and LineVisualizer: It is 2D Visualizer.
   LineVisualizer inherits Visualizer to avoid replicated codes.

9. Class PlayerControl: It is responsible for controlling Player.
    A. Update()
       : On Every frame it checks whether if the keyboard is pressed and moves the player.

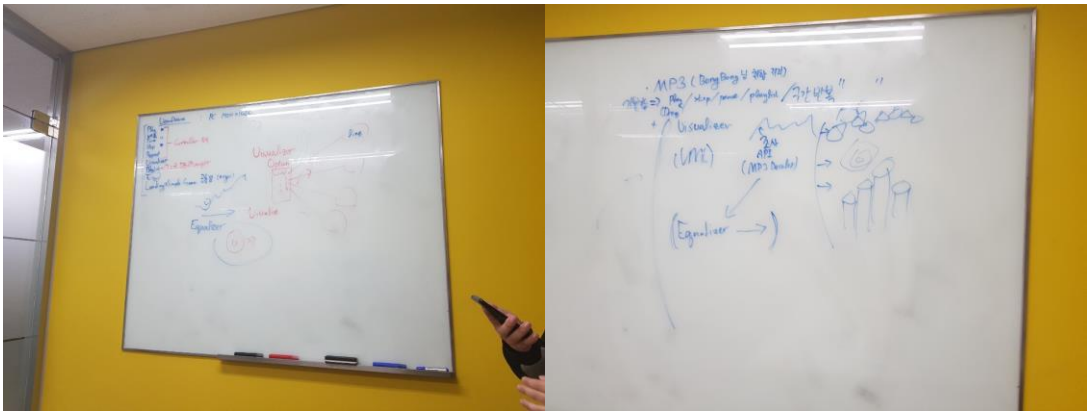
10. Class CameraControl: It is responsible for camera movement
    A. Update()
       : If Enter is pressed, Toggle mode to game or visualizer depending on state.
       If 'r' is pressed, Toggle watch mode mode to rotation or stop depending on state.

11. Class SpawnManager: It is responsible for enemy spawn.
    A. Hide()
       : Hides the enemy(the light).
    B. Show()
       : Shows the enemy.
    C. Warp()
       : Warps the enemy to new place.

12. Class EnemyControl: It is responsible for enemy's behavior.
    A. OnTriggerEnter(Collider)
       : If enemy is Caught by player, increase score and run away.

How we implemented

1. Drawn full picture of our system.



2. Listed all functions that we need to implement our program.
3. Designed all components that is responsible for one related system. (using UML)
4. Each team members implemented the independent components.
5. Slowly integrated all components into base of our program.

⇨ Important Implementation Issues

- UI Interaction:

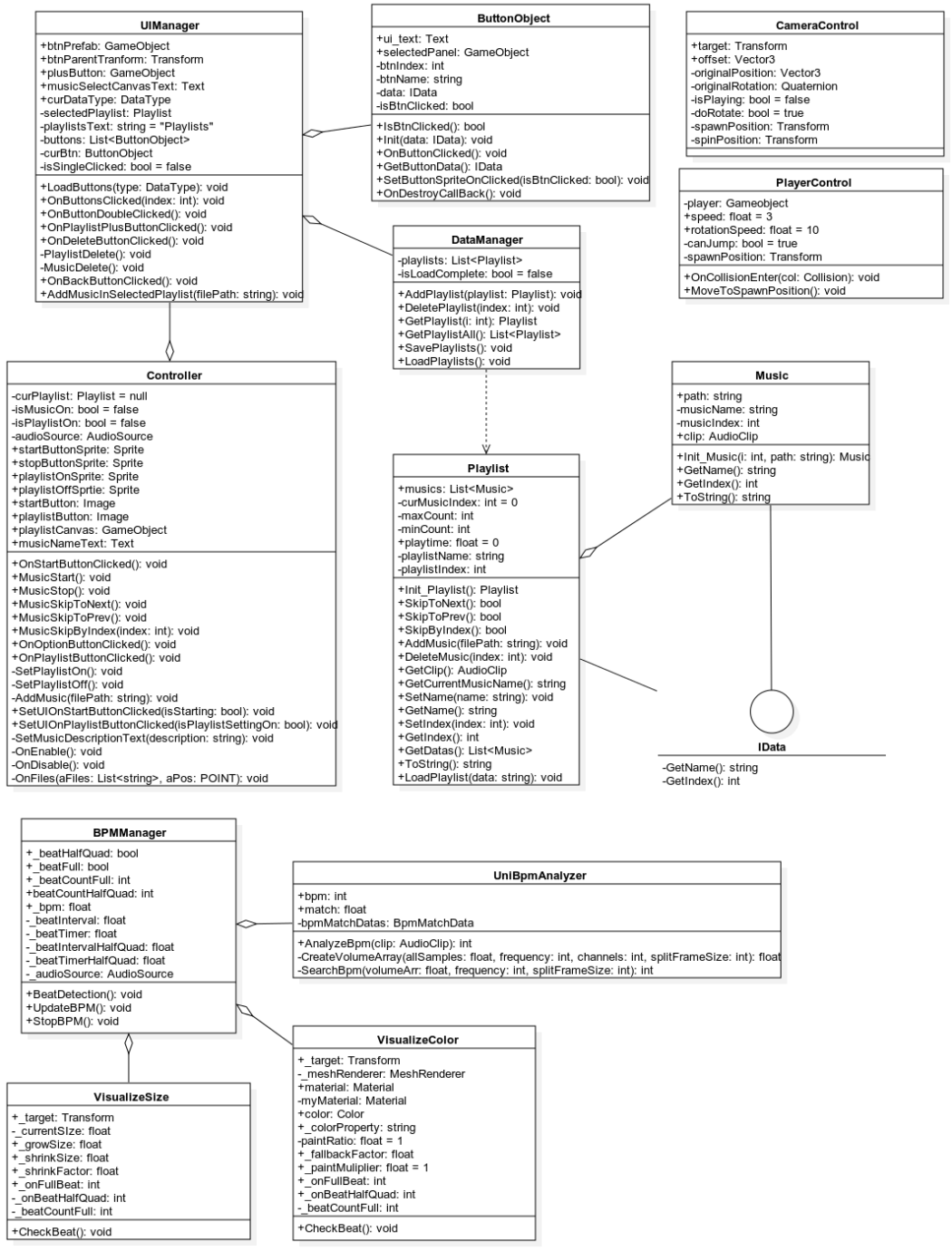
  Because there was many icons and button click. It was easy to make system dependent to each other. We had to think hard. We tried to give one responsibility to one object.

- Drag and Drop:

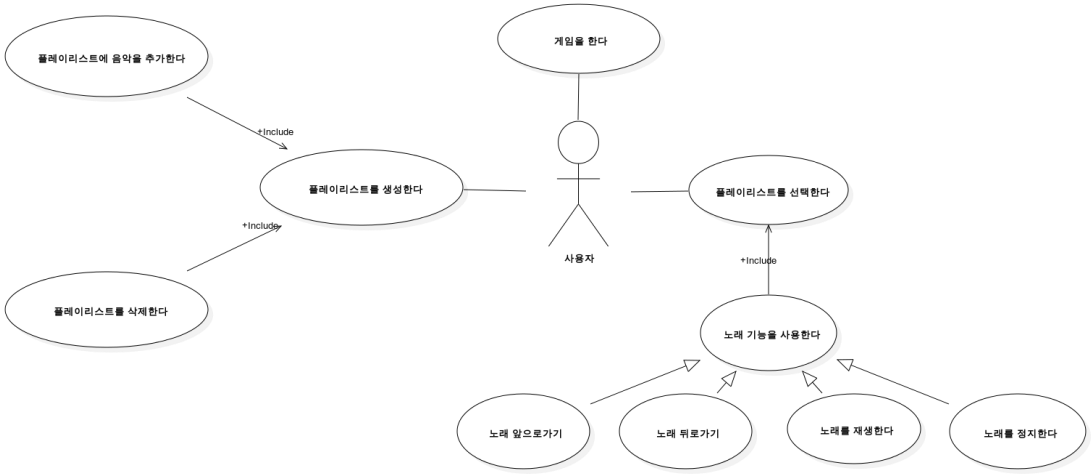
  Unity itself cannot handle drag and drop function. We tried everything with unity, but it was impossible. So, we had to use WinAPI for drag and drop function.

The result of UML modeling for system design:

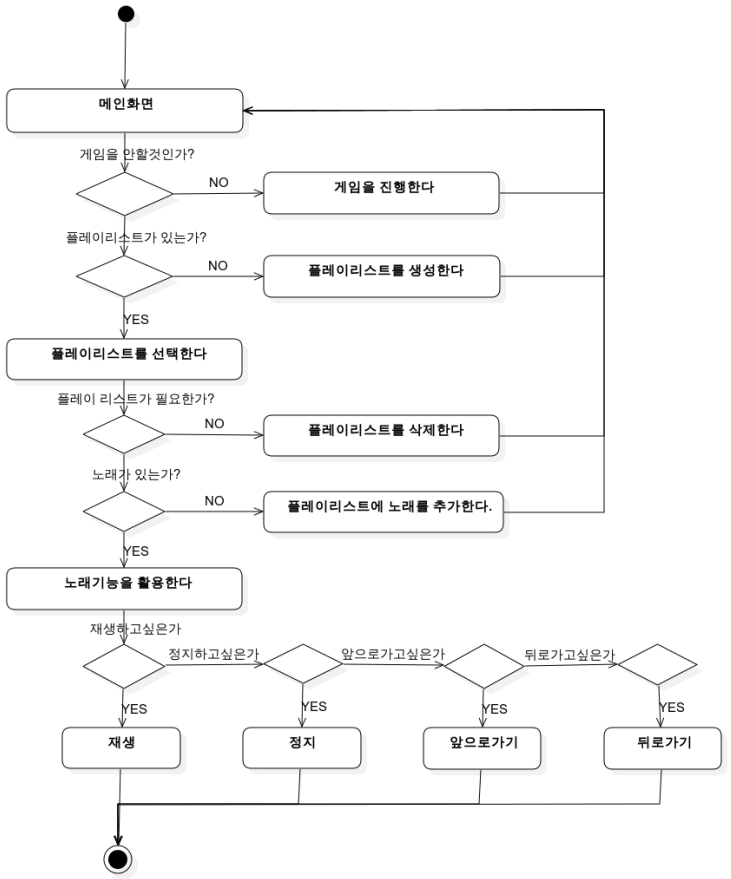
- Class Diagram



● Use Case Diagram

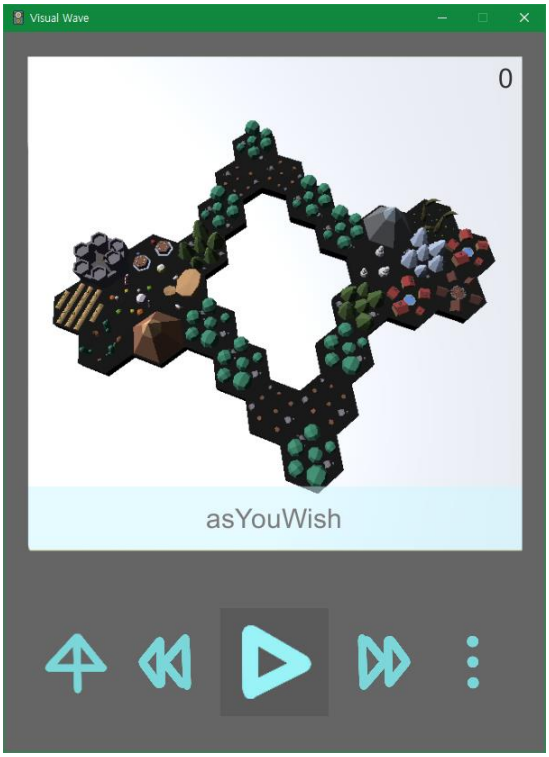


● Activity Diagram

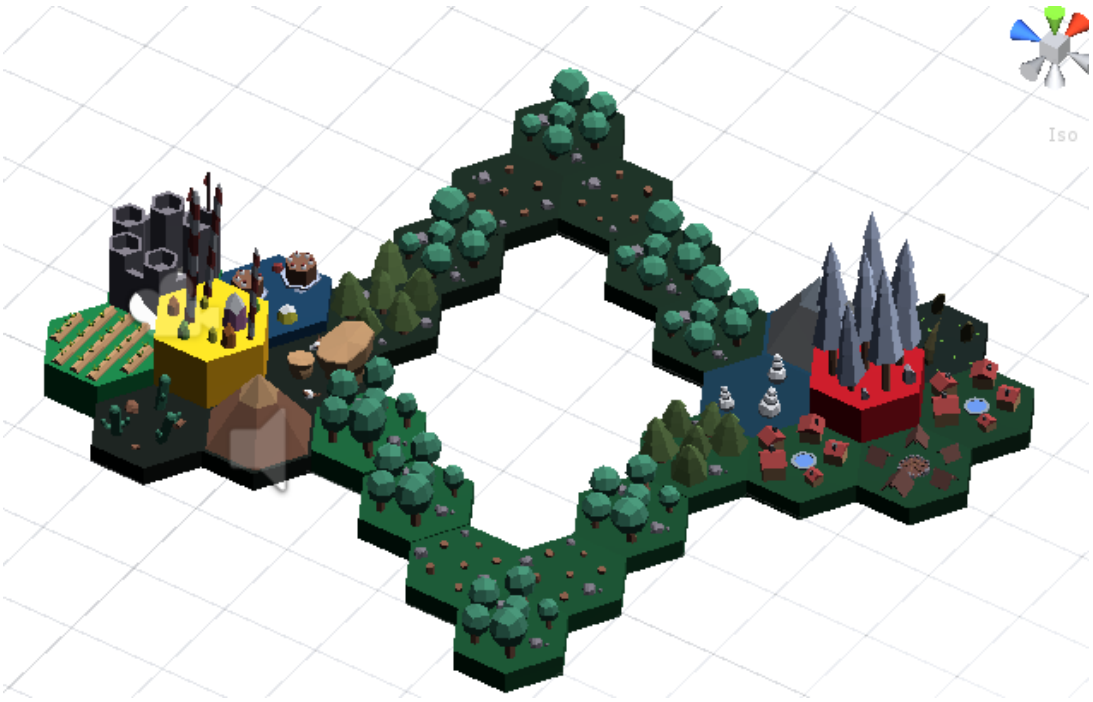


Execution Results:

1. MP3 UI



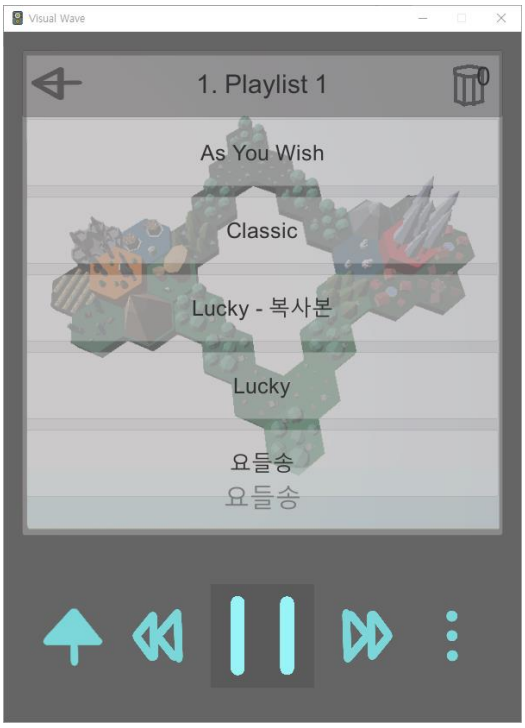
2. Visualizing Capture



3. Playlist

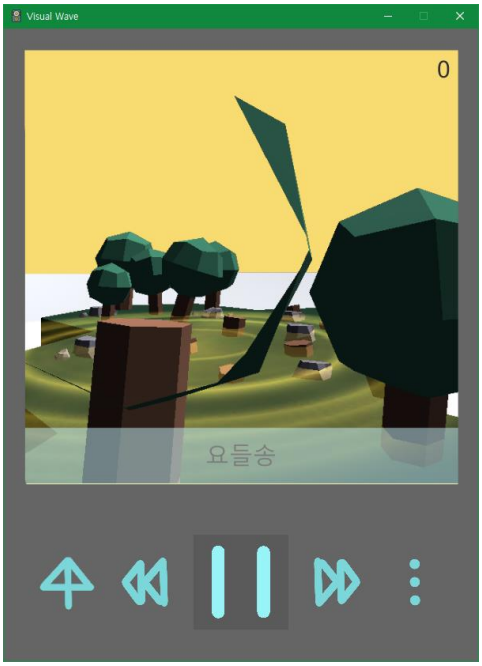
A. Make Playlist

B. Add Music into Playlist

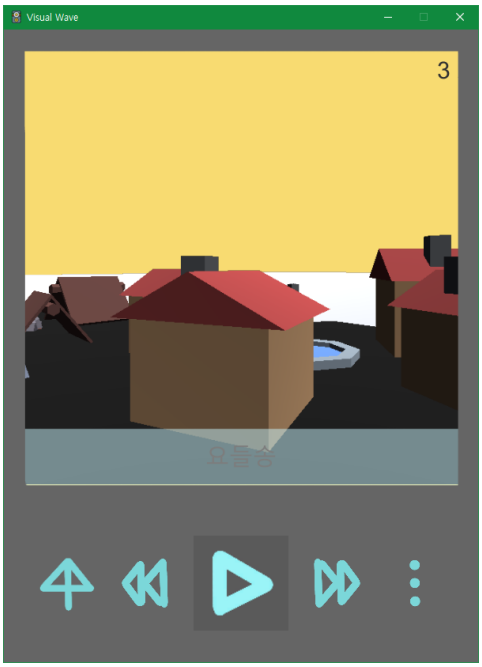


4. Executing (Playing Game)

A. Found Light Point



B. Earned 3 point!



- For more details, we recorded demo video. (In presentation folder)

How we applied object oriented concepts to the development for our project:

1. We made interface IData to handle various type of data in form of IData.
    A. This will make coupling weaker and make code more independent, reduce developing time.
    B. Interface IData

```
public interface IData
{
    string GetName();
    int GetIndex();
}
```

2. Playlist class and Music class are sharing IData interface. Both are used at UIManager class.
    A. Inherit Interface – Idata is implemented

```
public class Playlist : MonoBehaviour, IData
{
    public string GetName()
    {
        return playlistName;
    }

    public int GetIndex()
    {
        return playlistIndex;
    }

    .
    .
    .
}
```

3. Playlist class and Music class are overriding ToString function to print out string that fits itself.
    A. We can see polymorphism concept at overriding parts.
    B. Overriding ToString() function

```
public class Playlist : MonoBehaviour, IData
{
    public override string ToString()
    {
        //구분자는 '^'
        string playlistInfo = $"{playlistIndex}^{playlistName}^";

        int i = 0;
        foreach (var music in musics)
        {
            playlistInfo = $"{playlistInfo}{music.ToString()}";

            if (++i < musics.Count)
                playlistInfo = $"{playlistInfo}&";
        }

        return playlistInfo;
    }

    .
    .
    .
}
```

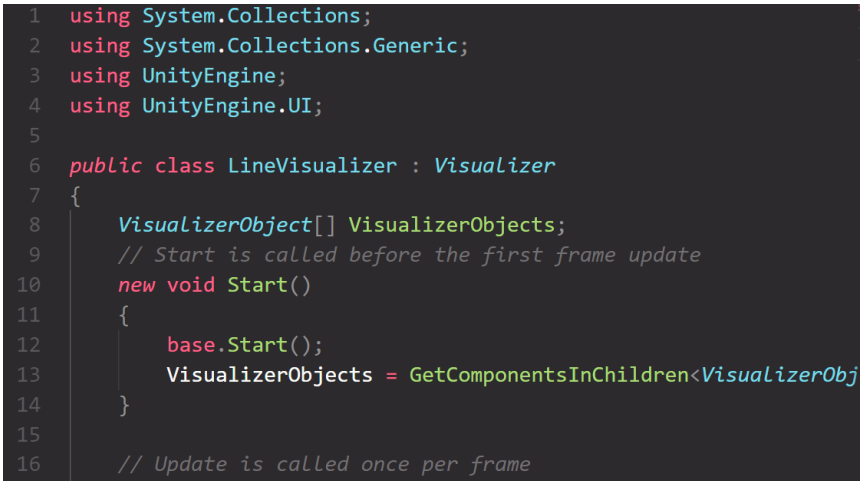
4. In most case of classes, we used getter and setter to give access restriction to each data. Information hiding was being kept.
    A. We used C#'s property which is shortcut for getter and setter.

```
private bool isBtnClicked = false;
public bool IsBtnClicked
{
    get { return isBtnClicked; }
    set
    {
        isBtnClicked = value;

        SetButtonSpriteOnClicked(isBtnClicked);
    }
}
```

5. Used Inheritance to avoid same code.

   A. Line Visualizer inherits Visualizer



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

public class LineVisualizer : Visualizer
{
    VisualizerObject[] VisualizerObjects;
    // Start is called before the first frame update
    new void Start()
    {
        base.Start();
        VisualizerObjects = GetComponentsInChildren<VisualizerObj
    }

    // Update is called once per frame
```

Conclusion (What we felt) :

- 고주형: As a team leader I lead the project. With the experience of project1, I can say that this project management was more successful and efficient. As I know the characteristic of all team members, I gave them appropriate work. All team members worked well with no cross talking. With this experience I somehow perceived team members as the objects working smartly with given message. This gave me feeling that OOP is very realistic like our life.
- 김호성: Interacting with various object was challenging for me. I spent a lot of time considering Single Responsibility Principle. It was helpful for me.
- 손희승: I used unity for the first time, but it was amazing. As C++ programmer, using another language concerning oop was new experience.
- 이준협: Using many kinds of UML I felt some insight of analyzing system. Drawing UML helps me to design large system. I would use these diagrams in another project.
- 이하람: It was a good project that I learned many things through communicating with my team members and doing it.