Culmination Project

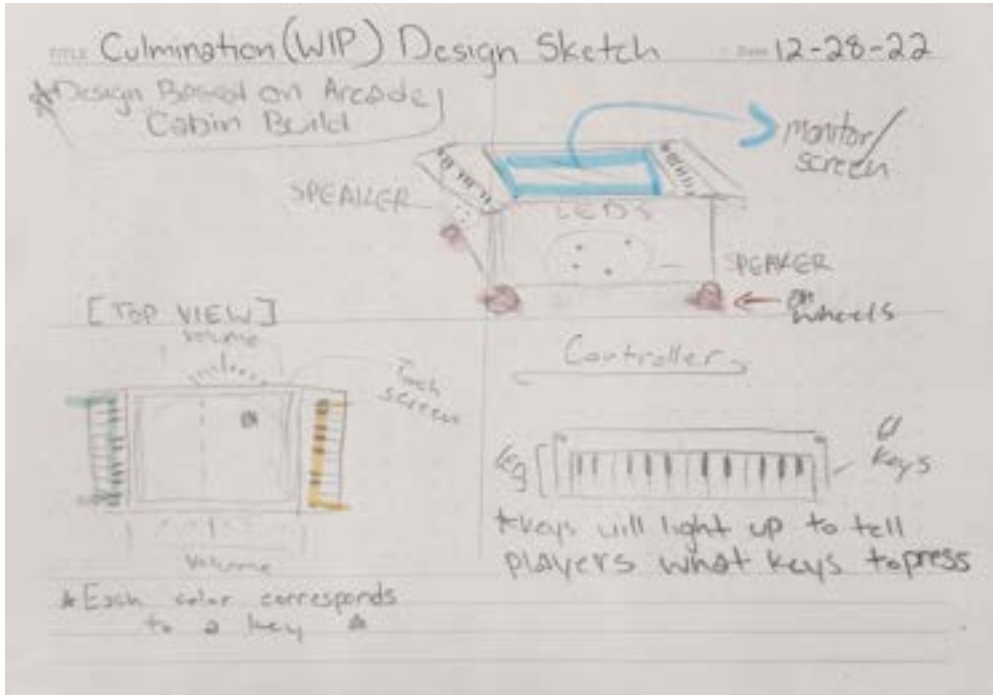
Lilia Torres

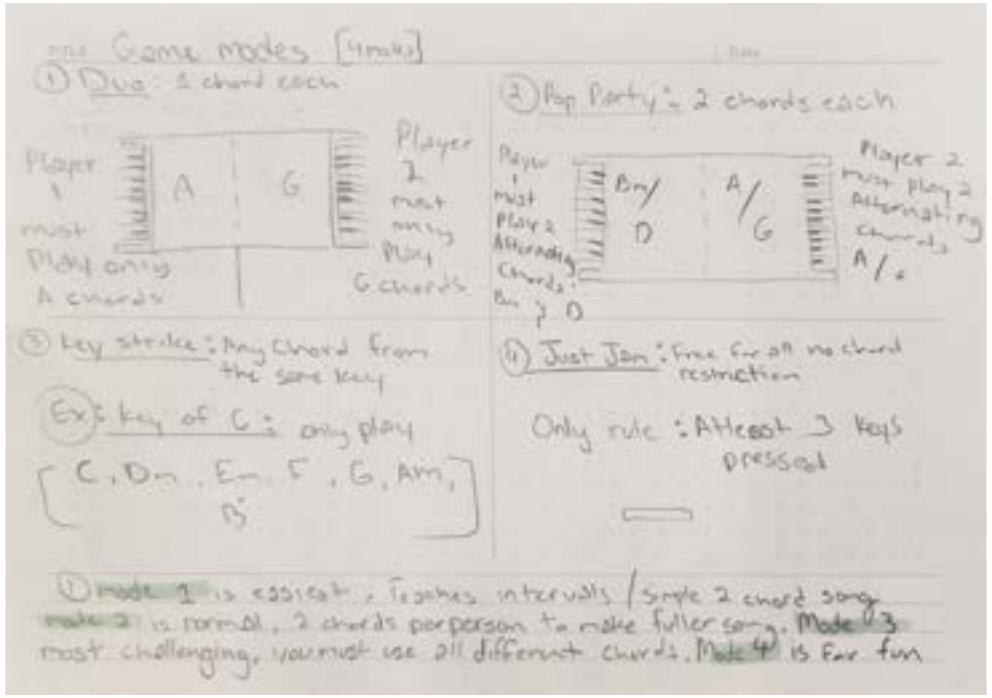
Introduction

Throughout my college experience I have explored the various ways audio can be more immersive and interactive.

Previously, my experience with audio has been related to music production, and sound design for mediums like, radio, podcast, and film and television.

For my culmination, I want to combine my experience with music, with my interest in video games, to create a fun and challenging twist to the Atari classic: Pong.

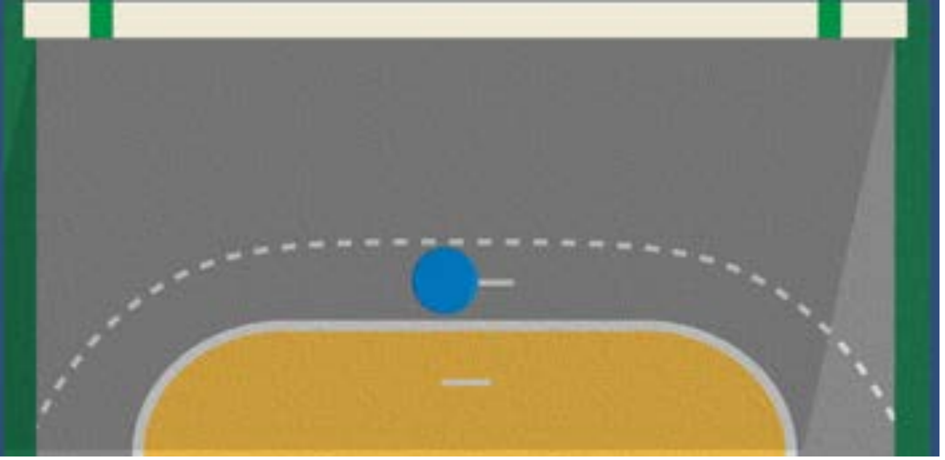

Culmination (WIP) Design Sketch
12-28-22
*Design Based on Arcade Cabin Build
monitor/screen
SPEAKER
LEDS
SPEAKER
on wheels
[TOP VIEW]
Volume
Touch screen
Volume
*Each color corresponds to a key *
Controller
Eg
keys
*keys will light up to tell players what keys to press

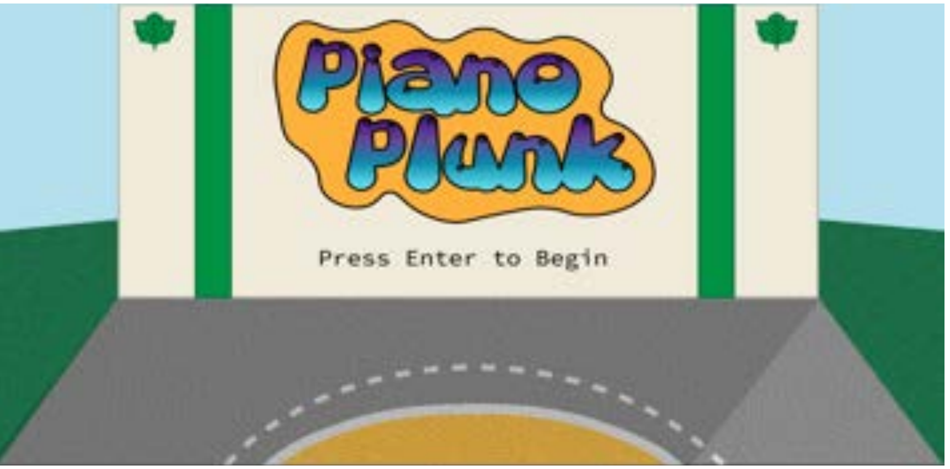

Game modes [4 modes]
① Duo: 1 chord each
Player 1 must play only A chords
A
G
Player 2 must only play G chords
② Pop Party: 2 chords each
Player 1 must play 2 alternating chords: Bm & D
Bm/D
A/G
Player 2 must play 2 alternating chords A/G
③ Key strike: Any chord from the same key
Ex: key of C: only play
[C, Dm, Em, F, G, Am, B°]
④ Just Jam: Free for all no chord restriction
Only rule: Atleast 3 keys pressed
① mode 1 is easiest, Teaches intervals / simple 2 chord song. mode 2 is normal, 2 chords per person to make fuller song. Mode 3 most challenging, you must use all different chords. Mode 4 is for fun

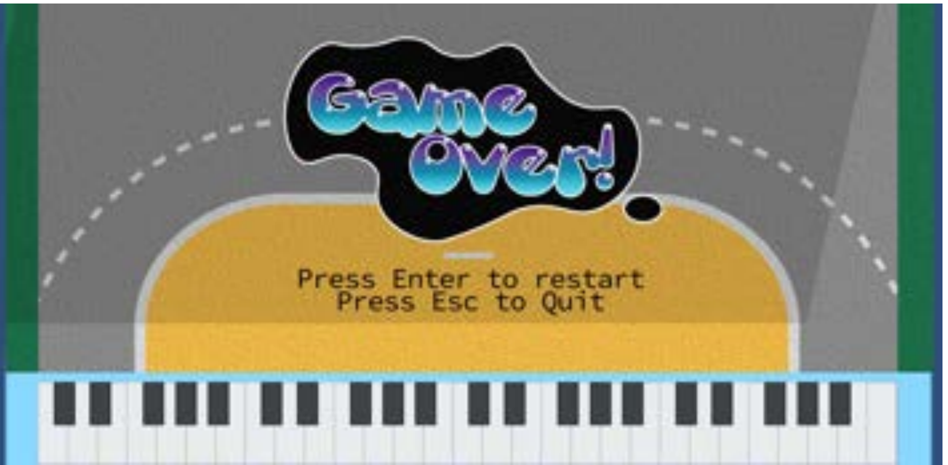
Project Description

Piano Plunk is table tennis with a musical twist. Players will have to play chords on a piano in order to hit the ball back and forth, creating a song.

The initial protype will be created in Unity, and I will use a variety of programs for the visuals and audio components of this project.





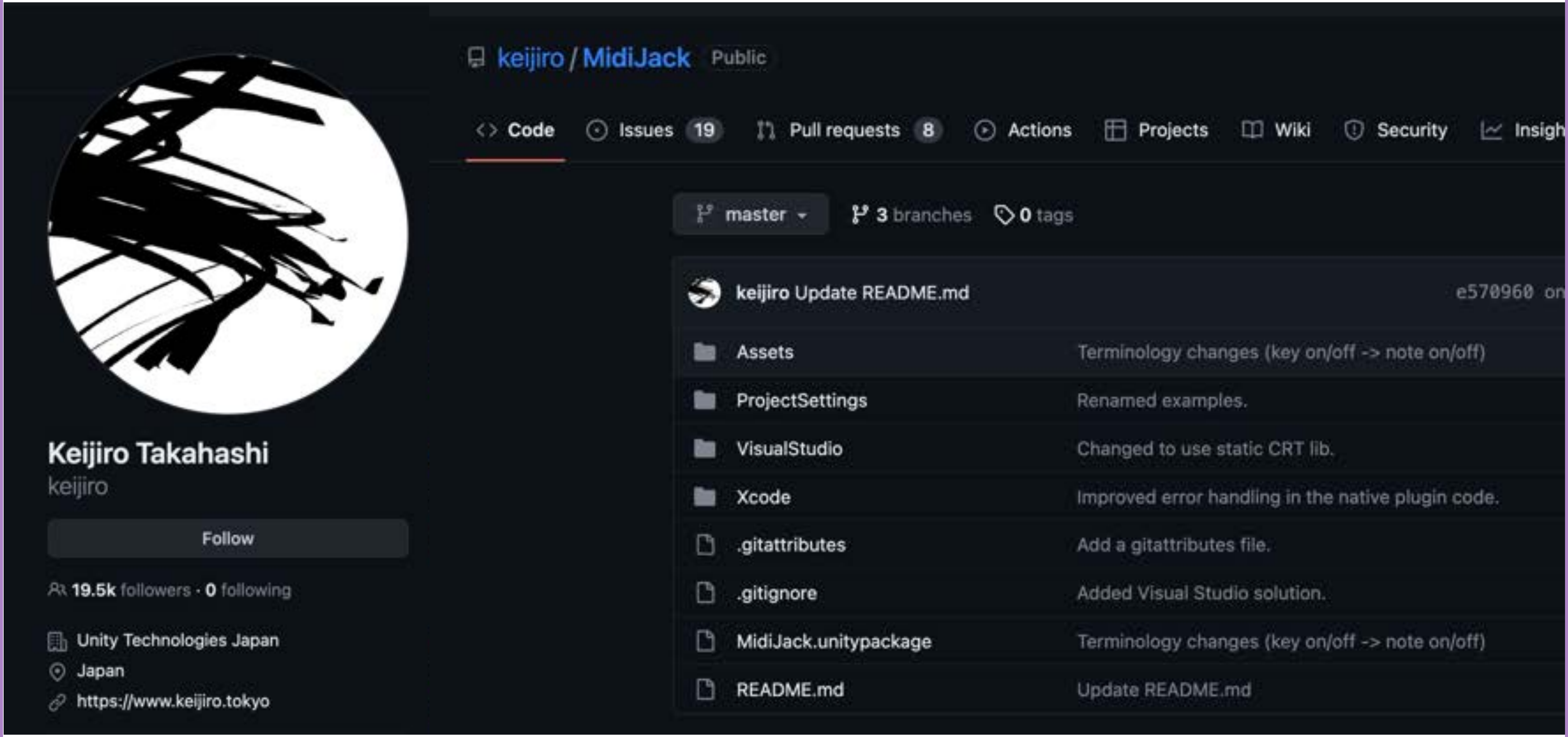


Main Concepts

Piano Pong will be a 1 to 2-player music game that teaches tonality and keys by challenging players to only use musical chords in order to hit the ball back and forth.

The player will have a piano on screen to guide them but will mostly be relying on memory and hand/finger placement to hit the incoming ball successfully.

All assets will be designed on paper, then brought to life through 4 main programs:



keijiro / MidiJack Public

Code · Issues 19 · Pull requests 8 · Actions · Projects · Wiki · Security · Insigh

master · 3 branches · 0 tags

keijiro Update README.md e570960 on

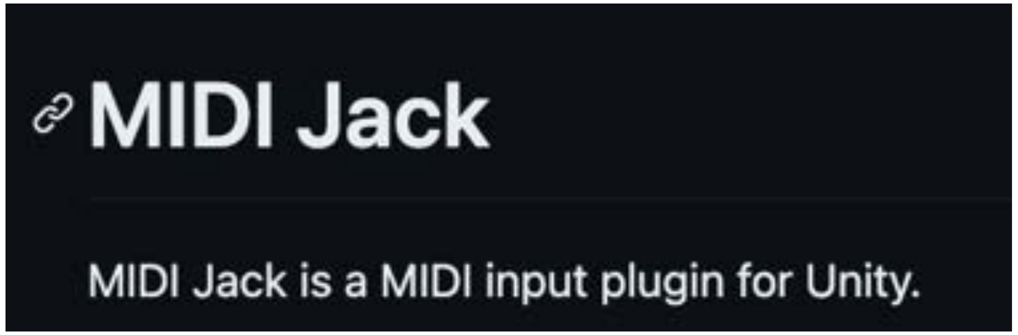
Assets	Terminology changes (key on/off -> note on/off)
ProjectSettings	Renamed examples.
VisualStudio	Changed to use static CRT lib.
Xcode	Improved error handling in the native plugin code.
.gitattributes	Add a gitattributes file.
.gitignore	Added Visual Studio solution.
MidiJack.unitypackage	Terminology changes (key on/off -> note on/off)
README.md	Update README.md

Keijiro Takahashi
keijiro

Follow

19.5k followers · 0 following

Unity Technologies Japan
Japan
https://www.keijiro.tokyo



MIDI Jack

MIDI Jack is a MIDI input plugin for Unity.

Keijiro Takahashi

Midi Controller

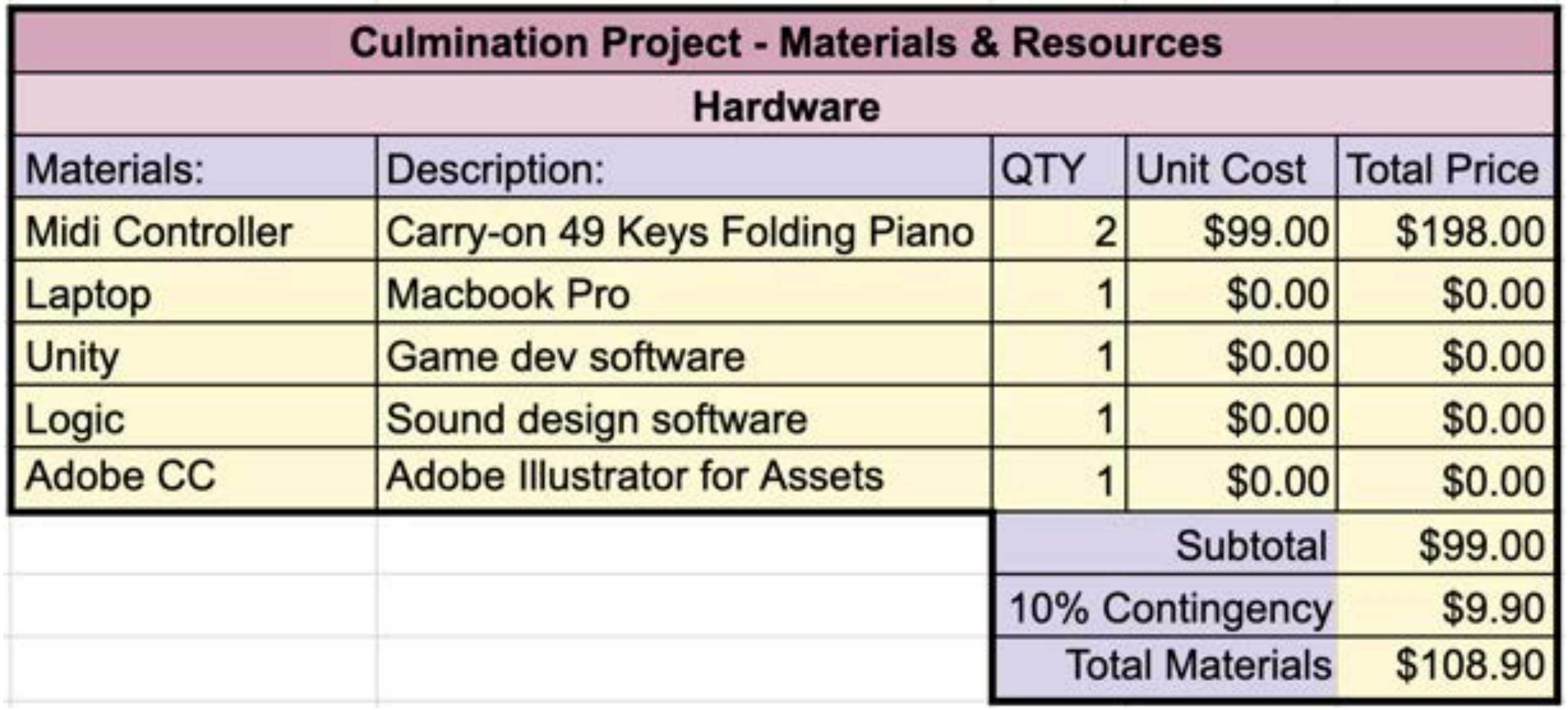
Positives

- 49 keys
- UBS Midi
- Foldable

Challenges

- Latency/Lag
- USB unreliable

Budget Estimate



Culmination Project - Materials & Resources				
Hardware				
Materials:	Description:	QTY	Unit Cost	Total Price
Midi Controller	Carry-on 49 Keys Folding Piano	2	$99.00	$198.00
Laptop	Macbook Pro	1	$0.00	$0.00
Unity	Game dev software	1	$0.00	$0.00
Logic	Sound design software	1	$0.00	$0.00
Adobe CC	Adobe Illustrator for Assets	1	$0.00	$0.00
			Subtotal	$99.00
			10% Contingency	$9.90
			Total Materials	$108.90

February

- ❑ Initial Planning
- ❑ Complete Game Document
- ❑ Budget Materials

March

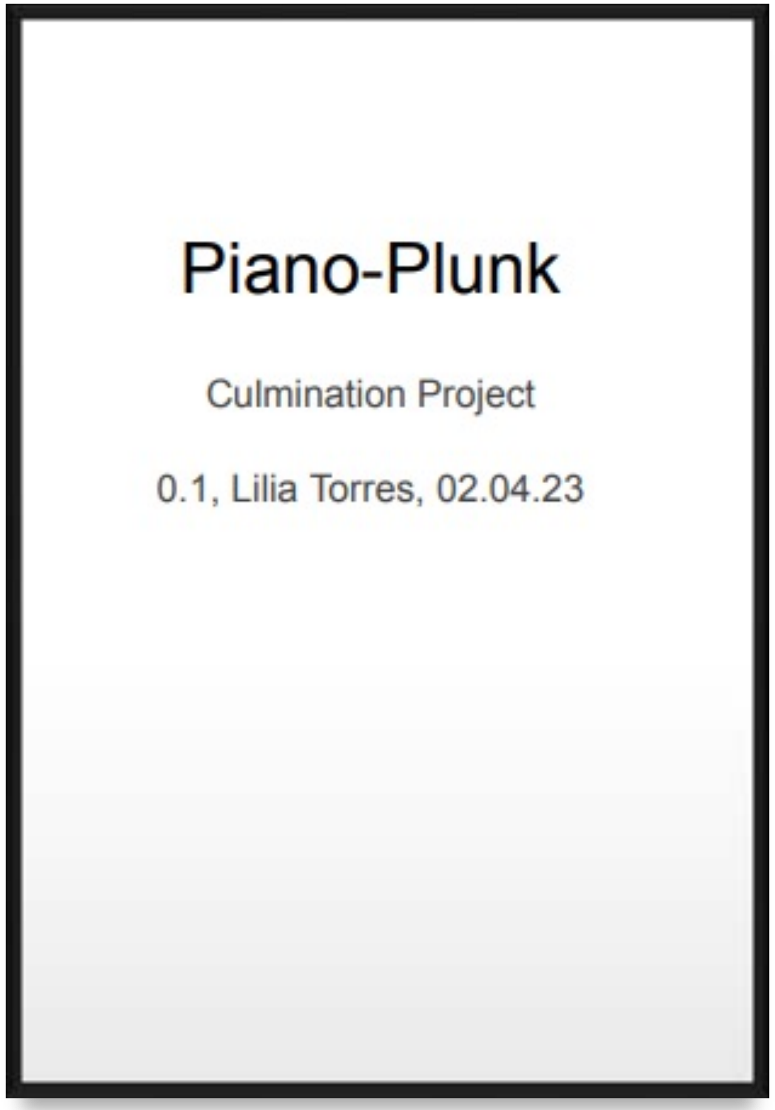
- ❑ Programing Game Physics
- ❑ Midi Implementation
- ❑ Asset Design

April

- ❑ Asset Implementation
- ❑ Sound Implementation
- ❑ Beta Testing

Monthly Goals – Game Plan

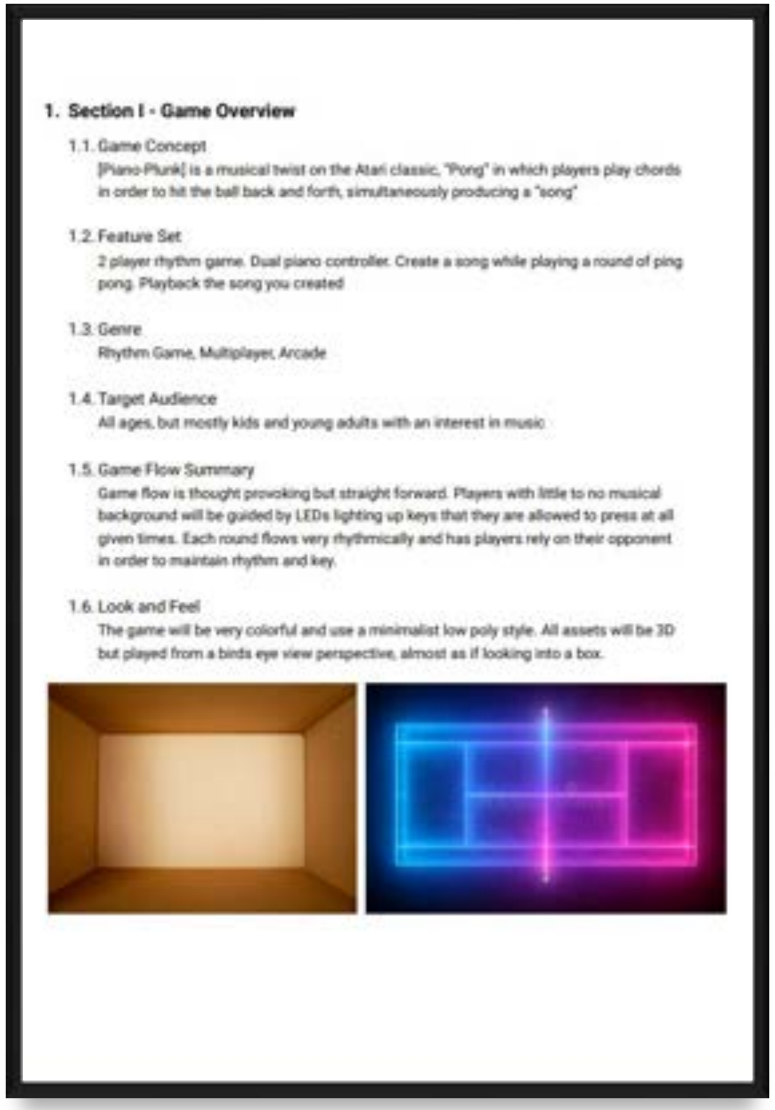
Deliverables



Piano-Plunk

Culmination Project

0.1, Lilia Torres, 02.04.23



1. **Section I - Game Overview**

1.1. Game Concept

[Piano-Plunk] is a musical twist on the Atari classic, "Pong" in which players play chords in order to hit the ball back and forth, simultaneously producing a "song"

1.2. Feature Set

2 player rhythm game. Dual piano controller. Create a song while playing a round of ping pong. Playback the song you created

1.3. Genre

Rhythm Game, Multiplayer, Arcade

1.4. Target Audience

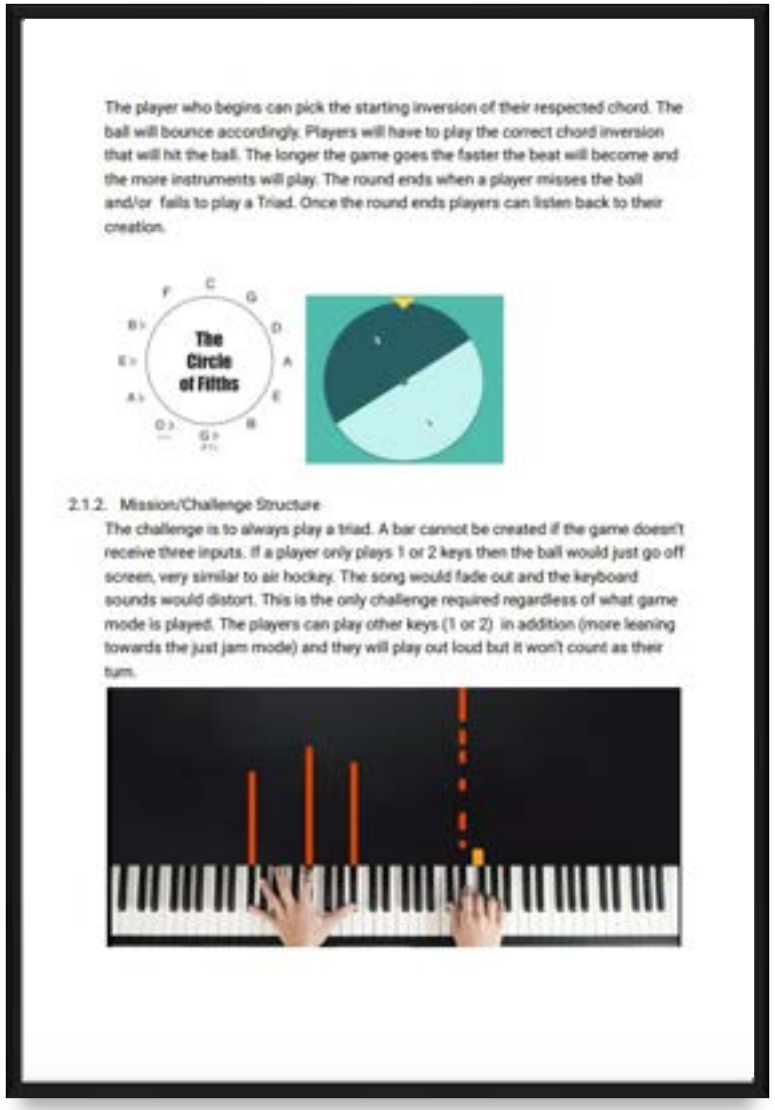
All ages, but mostly kids and young adults with an interest in music

1.5. Game Flow Summary

Game flow is thought provoking but straight forward. Players with little to no musical background will be guided by LEDs lighting up keys that they are allowed to press at all given times. Each round flows very rhythmically and has players rely on their opponent in order to maintain rhythm and key.

1.6. Look and Feel

The game will be very colorful and use a minimalist low poly style. All assets will be 3D but played from a birds eye view perspective, almost as if looking into a box.

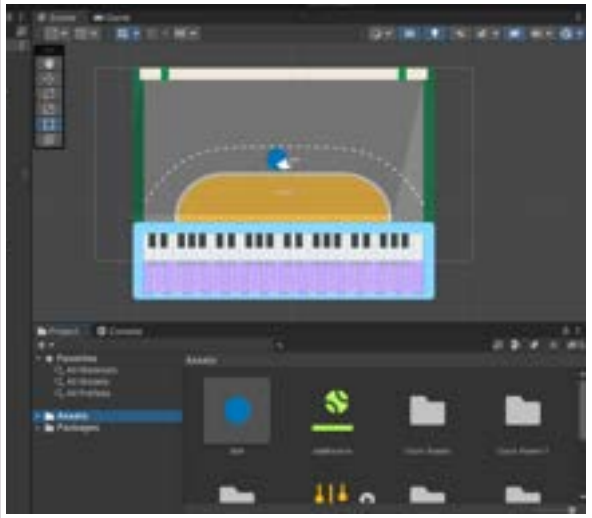


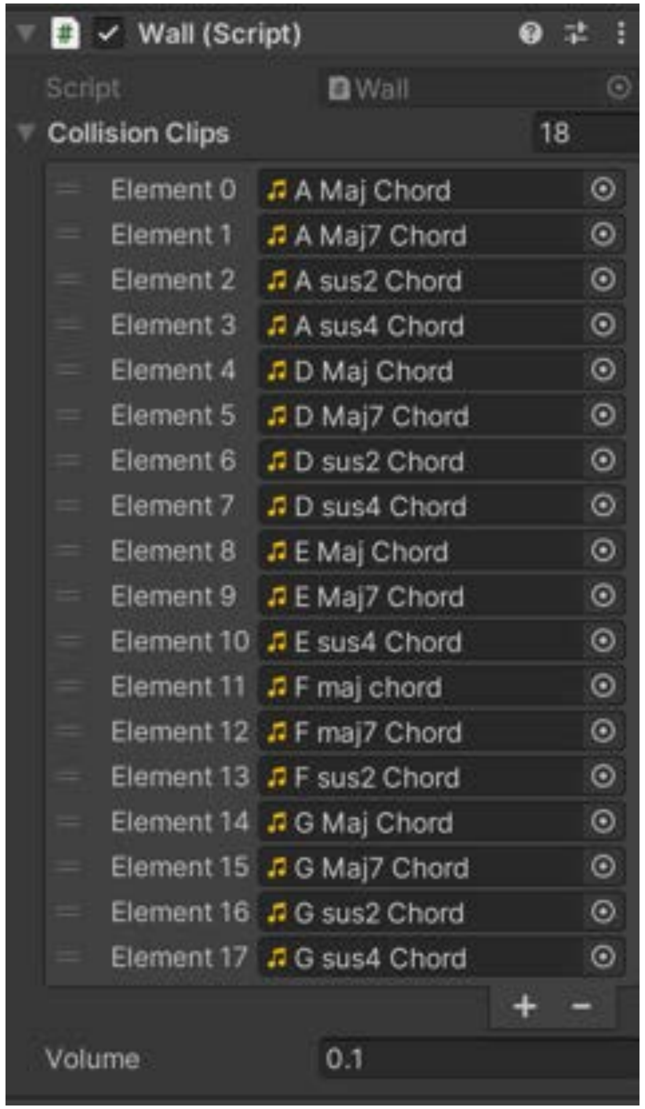
The player who begins can pick the starting inversion of their respected chord. The ball will bounce accordingly. Players will have to play the correct chord inversion that will hit the ball. The longer the game goes the faster the beat will become and the more instruments will play. The round ends when a player misses the ball and/or fails to play a Triad. Once the round ends players can listen back to their creation.

2.1.2. Mission/Challenge Structure

The challenge is to always play a triad. A bar cannot be created if the game doesn't receive three inputs. If a player only plays 1 or 2 keys then the ball would just go off screen, very similar to air hockey. The song would fade out and the keyboard sounds would distort. This is the only challenge required regardless of what game mode is played. The players can play other keys (1 or 2) in addition (more leaning towards the just jam mode) and they will play out loud but it won't count as their turn.

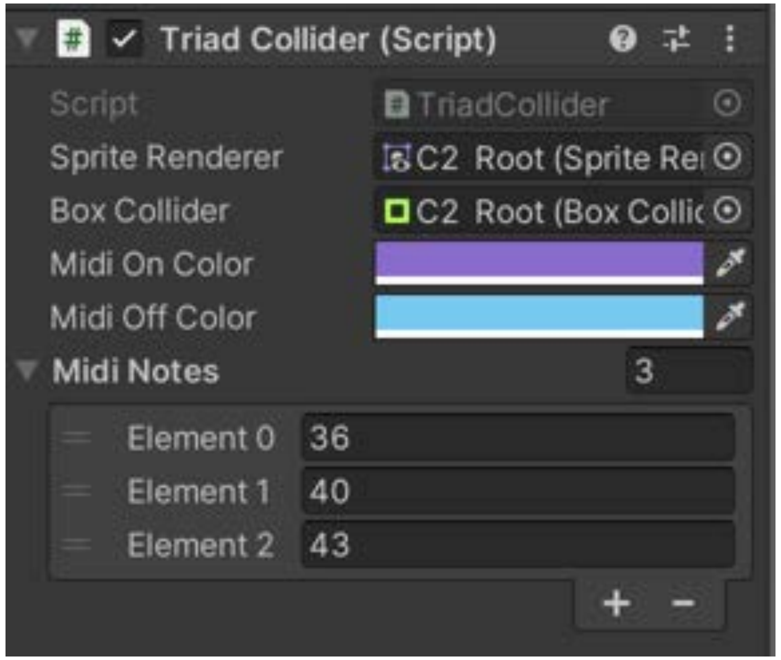
PIANO PLUNK PROTOTYPE





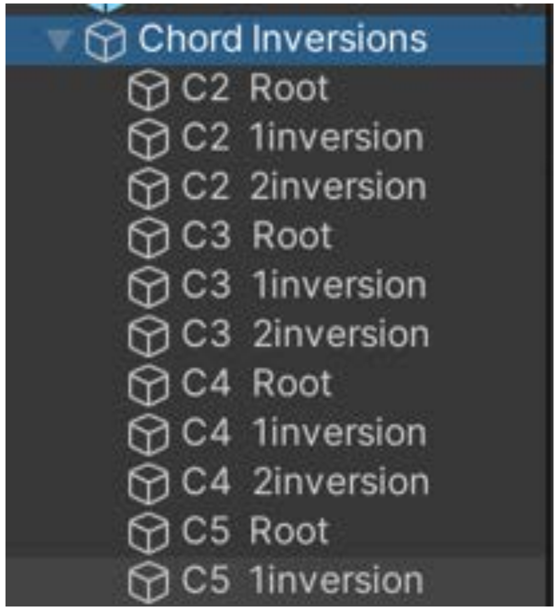
Wall (Script)

Script	Wall
Collision Clips	18
Element 0	A Maj Chord
Element 1	A Maj7 Chord
Element 2	A sus2 Chord
Element 3	A sus4 Chord
Element 4	D Maj Chord
Element 5	D Maj7 Chord
Element 6	D sus2 Chord
Element 7	D sus4 Chord
Element 8	E Maj Chord
Element 9	E Maj7 Chord
Element 10	E sus4 Chord
Element 11	F maj chord
Element 12	F maj7 Chord
Element 13	F sus2 Chord
Element 14	G Maj Chord
Element 15	G Maj7 Chord
Element 16	G sus2 Chord
Element 17	G sus4 Chord
Volume	0.1



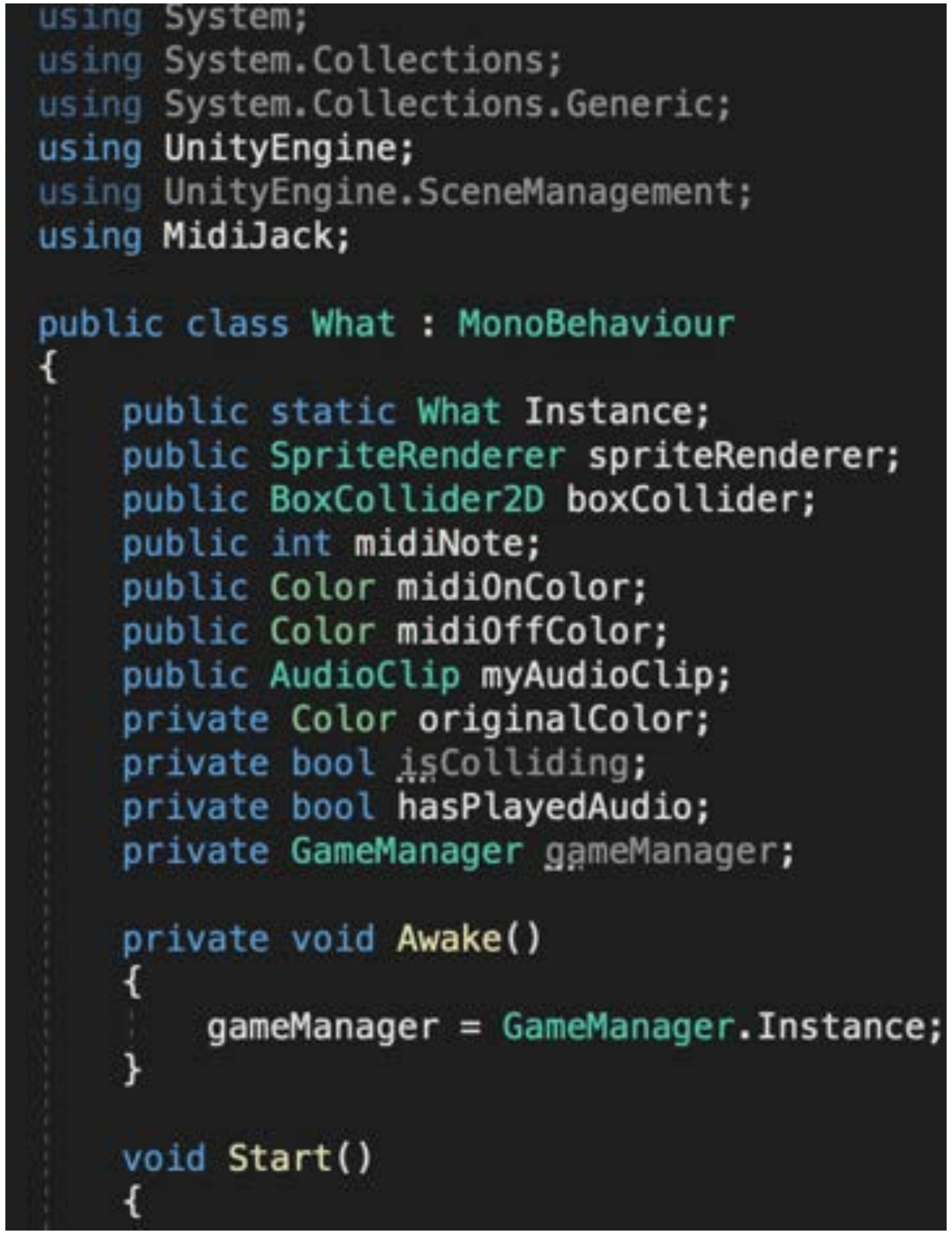
Triad Collider (Script)

Script	TriadCollider
Sprite Renderer	C2 Root (Sprite Rei
Box Collider	C2 Root (Box Collic
Midi On Color	🟪
Midi Off Color	🟦
Midi Notes	3
Element 0	36
Element 1	40
Element 2	43

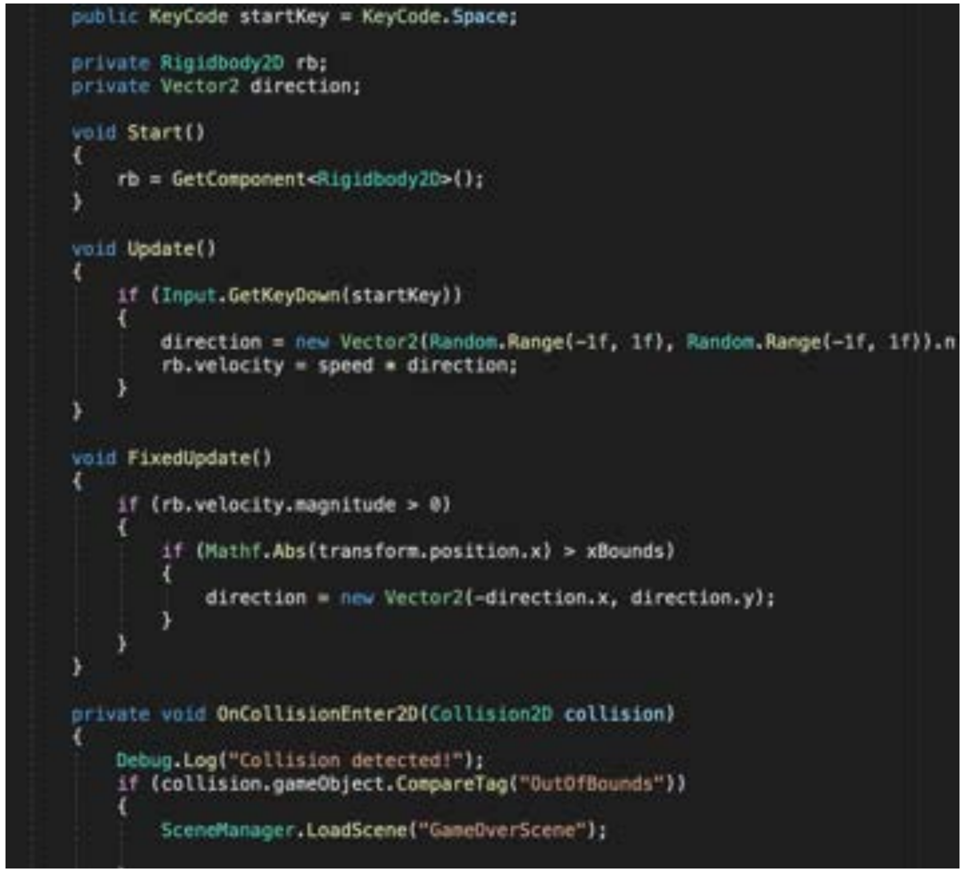


Chord Inversions
- C2 Root
- C2 1inversion
- C2 2inversion
- C3 Root
- C3 1inversion
- C3 2inversion
- C4 Root
- C4 1inversion
- C4 2inversion
- C5 Root
- C5 1inversion

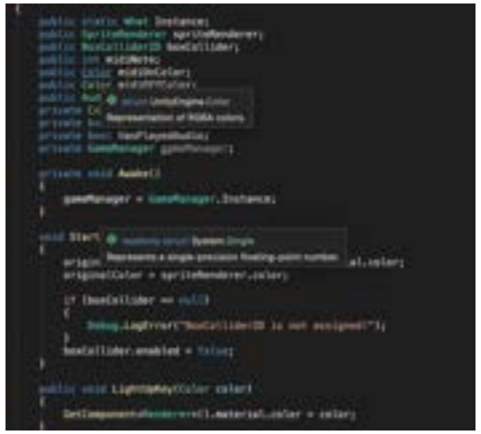
Main Scripting Components



```csharp
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using MidiJack;

public class What : MonoBehaviour
{
    public static What Instance;
    public SpriteRenderer spriteRenderer;
    public BoxCollider2D boxCollider;
    public int midiNote;
    public Color midiOnColor;
    public Color midiOffColor;
    public AudioClip myAudioClip;
    private Color originalColor;
    private bool isColliding;
    private bool hasPlayedAudio;
    private GameManager gameManager;

    private void Awake()
    {
        gameManager = GameManager.Instance;
    }

    void Start()
    {
```

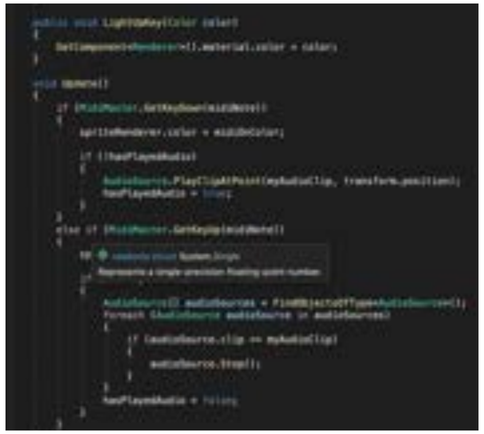


```csharp
public KeyCode startKey = KeyCode.Space;

private Rigidbody2D rb;
private Vector2 direction;

void Start()
{
    rb = GetComponent<Rigidbody2D>();
}

void Update()
{
    if (Input.GetKeyDown(startKey))
    {
        direction = new Vector2(Random.Range(-1f, 1f), Random.Range(-1f, 1f)).n
        rb.velocity = speed * direction;
    }
}

void FixedUpdate()
{
    if (rb.velocity.magnitude > 0)
    {
        if (Mathf.Abs(transform.position.x) > xBounds)
        {
            direction = new Vector2(-direction.x, direction.y);
        }
    }
}

private void OnCollisionEnter2D(Collision2D collision)
{
    Debug.Log("Collision detected!");
    if (collision.gameObject.CompareTag("OutOfBounds"))
    {
        SceneManager.LoadScene("GameOverScene");
```



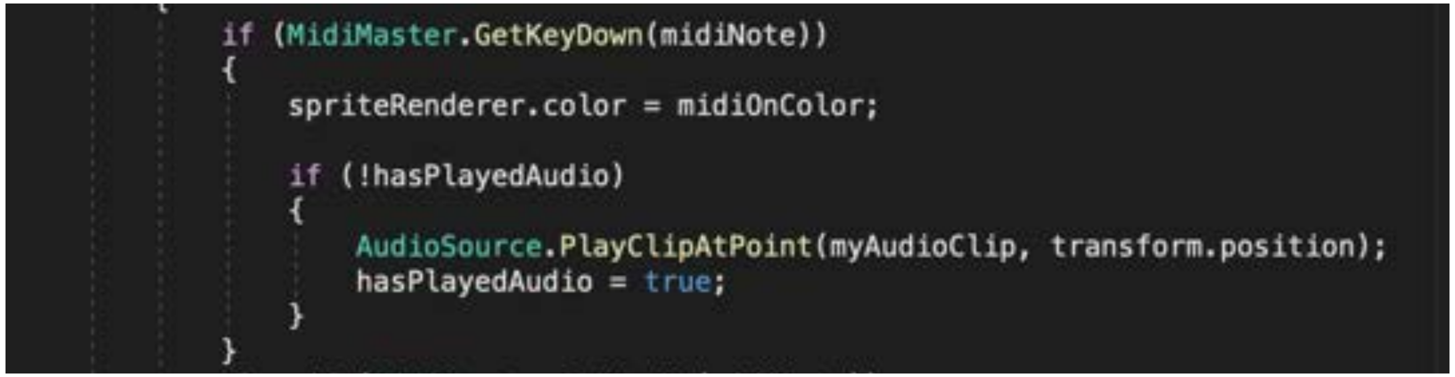
```csharp
{
    public static What Instance;
    public SpriteRenderer spriteRenderer;
    public BoxCollider2D boxCollider;
    public int midiNote;
    public Color midiOnColor;
    public Color midiOffColor;
    public AudioClip myAudioClip;
    private Color originalColor;
    private bool isColliding;
    private bool hasPlayedAudio;
    private GameManager gameManager;

    private void Awake()
    {
        gameManager = GameManager.Instance;
    }

    void Start()
    {
        origin[illegible]al.color;
        originalColor = spriteRenderer.color;

        if (boxCollider == null)
        {
            Debug.LogError("BoxCollider2D is not assigned!");
        }
        boxCollider.enabled = false;
    }

    public void LightUpKey(Color color)
    {
        GetComponent<Renderer>().material.color = color;
    }
```

struct UnityEngine.Color
Representation of RGBA colors.
readonly struct System.Single
Represents a single-precision floating-point number.

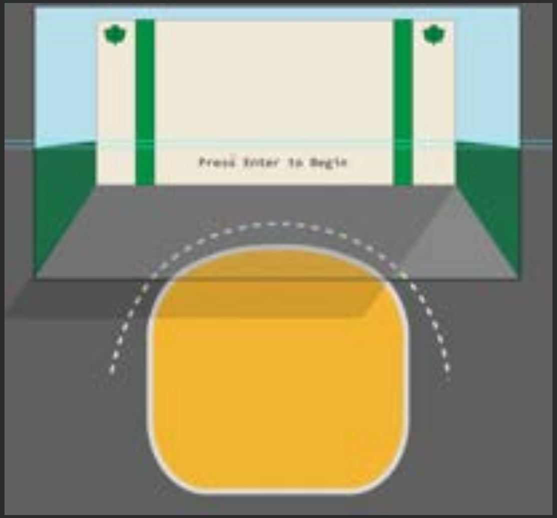


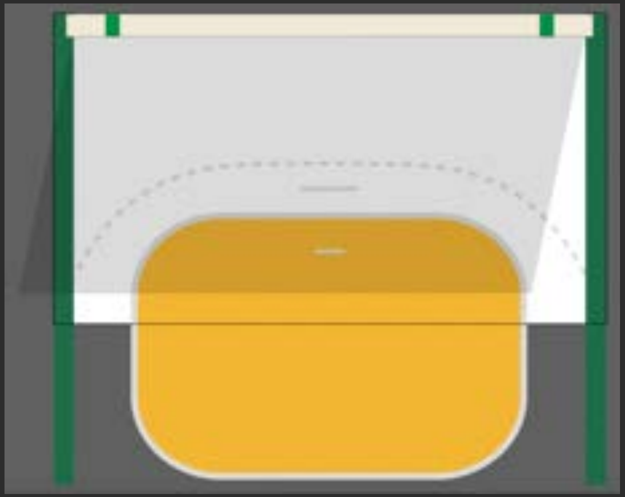
```csharp
public void LightUpKey(Color color)
{
    GetComponent<Renderer>().material.color = color;
}

void Update()
{
    if (MidiMaster.GetKeyDown(midiNote))
    {
        spriteRenderer.color = midiOnColor;

        if (!hasPlayedAudio)
        {
            AudioSource.PlayClipAtPoint(myAudioClip, transform.position);
            hasPlayedAudio = true;
        }
    }
    else if (MidiMaster.GetKeyUp(midiNote))
    {
        [illegible]
        {
            AudioSource[] audioSources = FindObjectsOfType<AudioSource>();
            foreach (AudioSource audioSource in audioSources)
            {
                if (audioSource.clip == myAudioClip)
                {
                    audioSource.Stop();
                }
            }
            hasPlayedAudio = false;
        }
    }
```

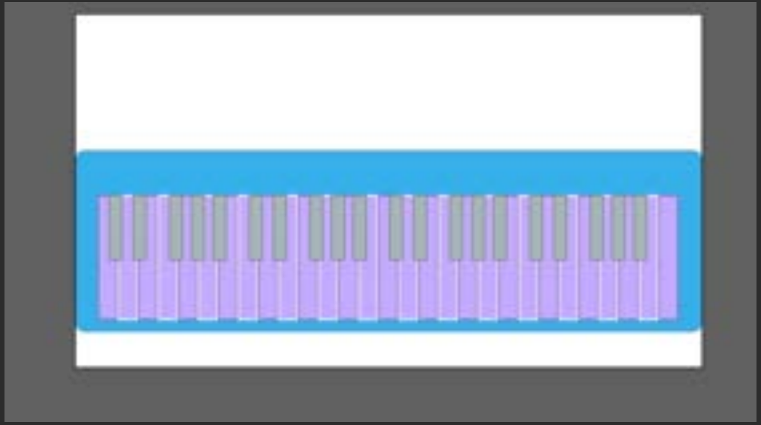
readonly struct System.Single
Represents a single-precision floating-point number.



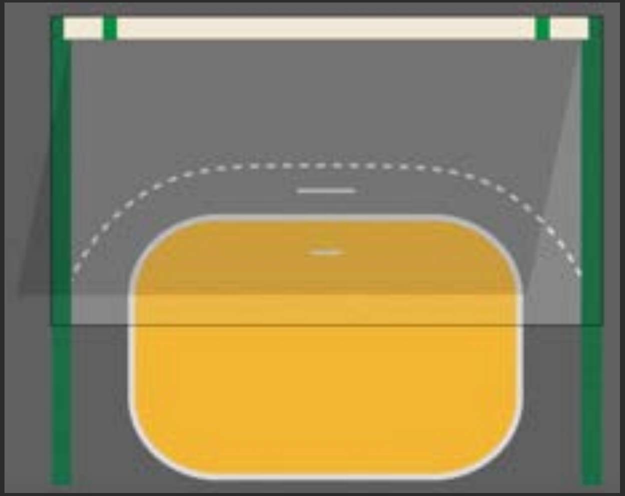
```csharp
        if (MidiMaster.GetKeyDown(midiNote))
        {
            spriteRenderer.color = midiOnColor;

            if (!hasPlayedAudio)
            {
                AudioSource.PlayClipAtPoint(myAudioClip, transform.position);
                hasPlayedAudio = true;
            }
        }
```

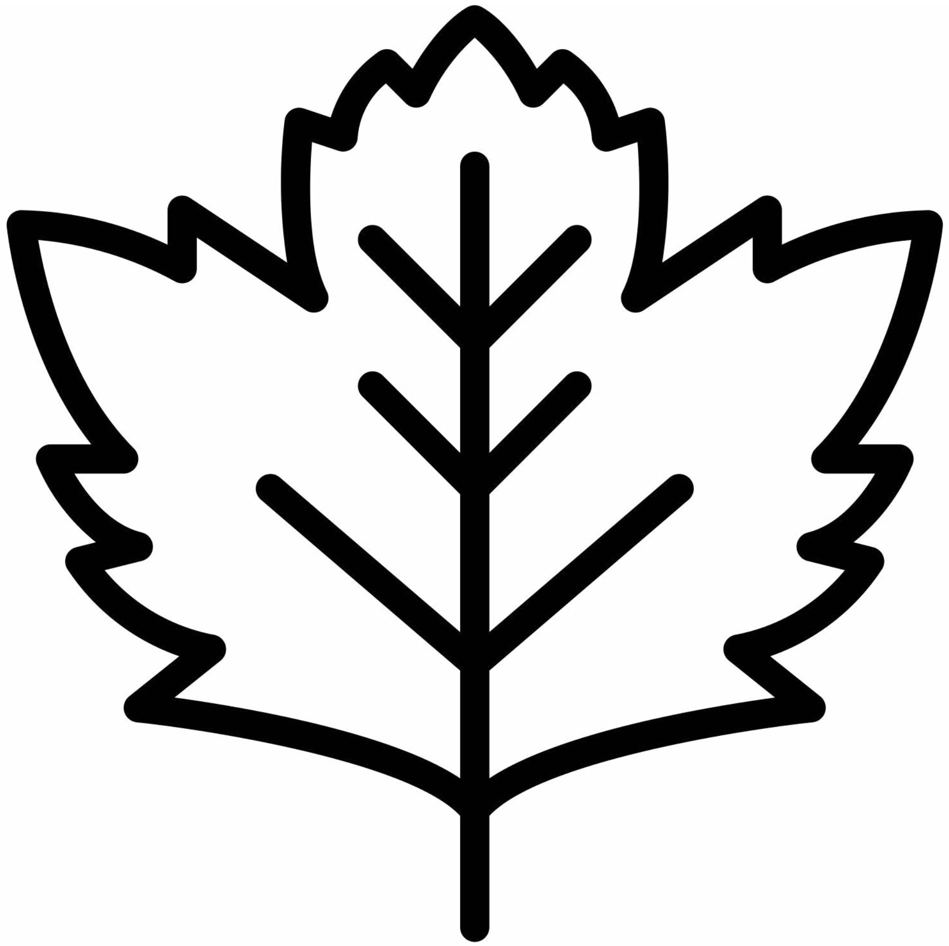

Press Enter to Begin





Piano
Plunk





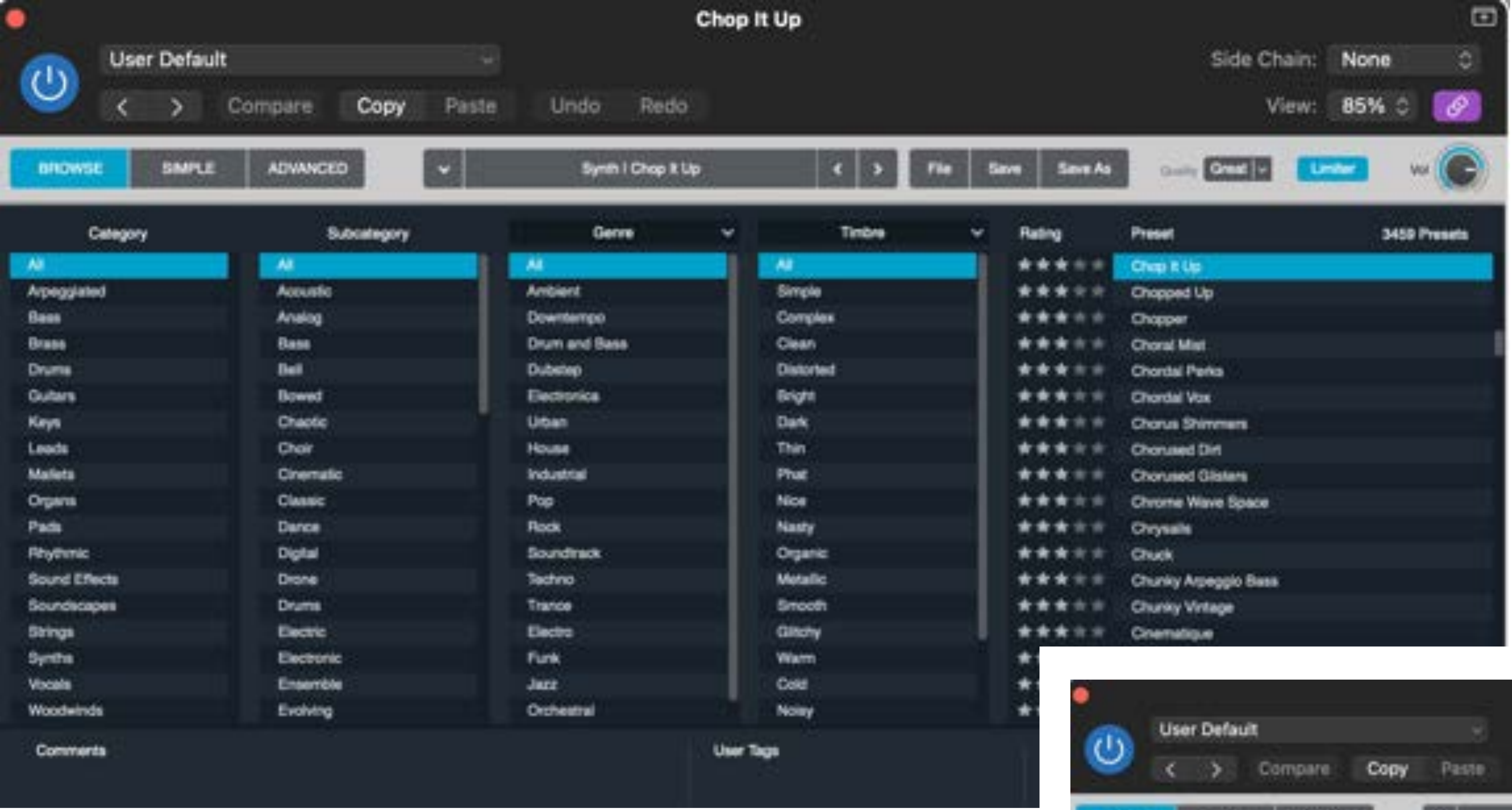
Created by tulpahn

Game Over!

Credits: Font Meme - Graffiti Creator

Piano Plunk

Credits: Font Meme - Graffiti Creator



Chop It Up

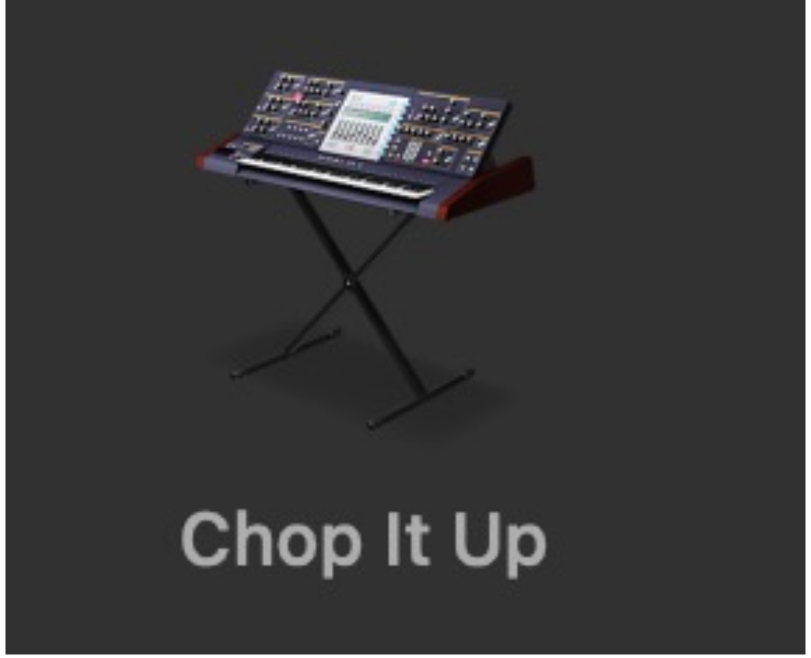
User Default | Side Chain: None | View: 85%

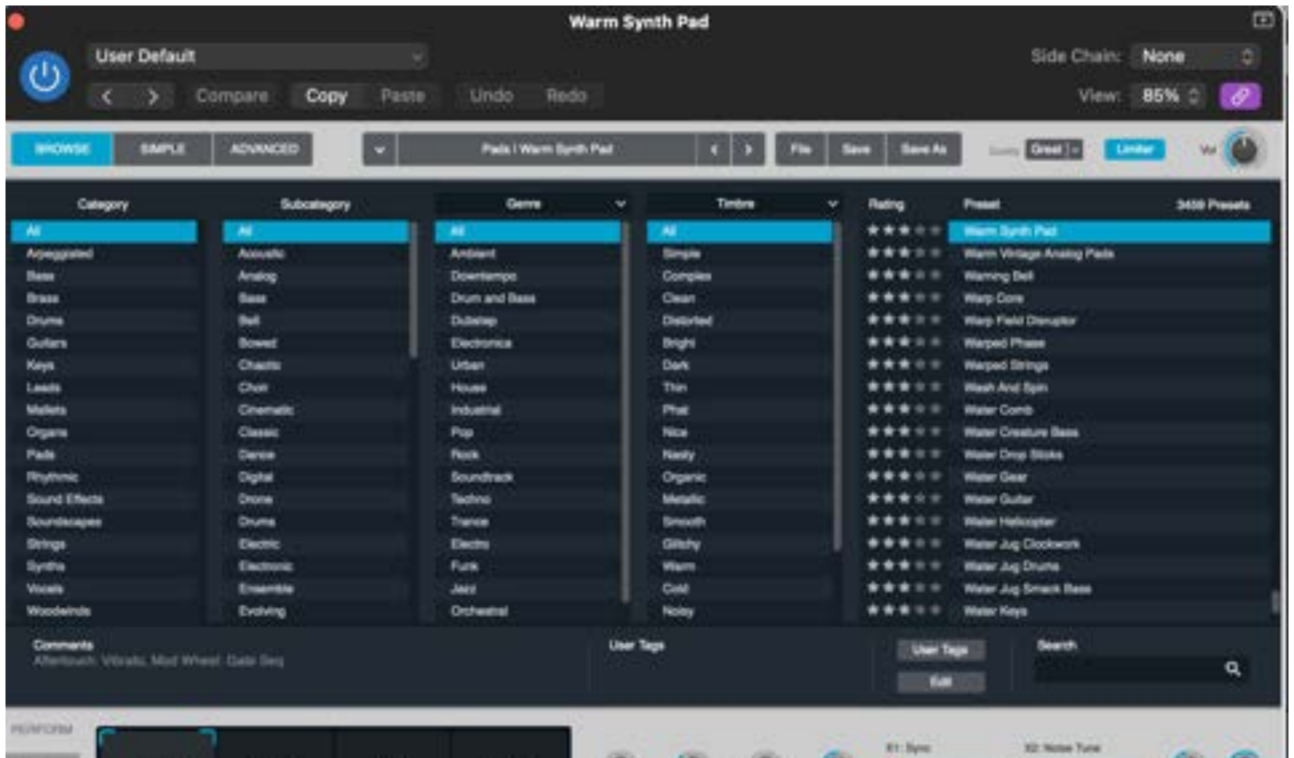
Compare · Copy · Paste · Undo · Redo

BROWSE · SIMPLE · ADVANCED | Synth | Chop It Up | File · Save · Save As | Quality: Great | Limiter | Vol

Category	Subcategory	Genre	Timbre	Rating	Preset (3459 Presets)
All	All	All	All	★★★☆☆	Chop It Up
Arpeggiated	Acoustic	Ambient	Simple	★★★☆☆	Chopped Up
Bass	Analog	Downtempo	Complex	★★★☆☆	Chopper
Brass	Bass	Drum and Bass	Clean	★★★☆☆	Choral Mist
Drums	Bell	Dubstep	Distorted	★★★☆☆	Chordal Perks
Guitars	Bowed	Electronica	Bright	★★★☆☆	Chordal Vox
Keys	Chaotic	Urban	Dark	★★★☆☆	Chorus Shimmers
Leads	Choir	House	Thin	★★★☆☆	Chorused Dirt
Mallets	Cinematic	Industrial	Phat	★★★☆☆	Chorused Glisters
Organs	Classic	Pop	Nice	★★★☆☆	Chrome Wave Space
Pads	Dance	Rock	Nasty	★★★☆☆	Chrysalis
Rhythmic	Digital	Soundtrack	Organic	★★★☆☆	Chuck
Sound Effects	Drone	Techno	Metallic	★★★☆☆	Chunky Arpeggio Bass
Soundscapes	Drums	Trance	Smooth	★★★☆☆	Chunky Vintage
Strings	Electric	Electro	Glitchy	★★★☆☆	Cinematique
Synths	Electronic	Funk	Warm	★	
Vocals	Ensemble	Jazz	Cold	★	
Woodwinds	Evolving	Orchestral	Noisy	★	

Comments | User Tags





Warm Synth Pad

User Default | Side Chain: None | View: 85%

Compare · Copy · Paste · Undo · Redo

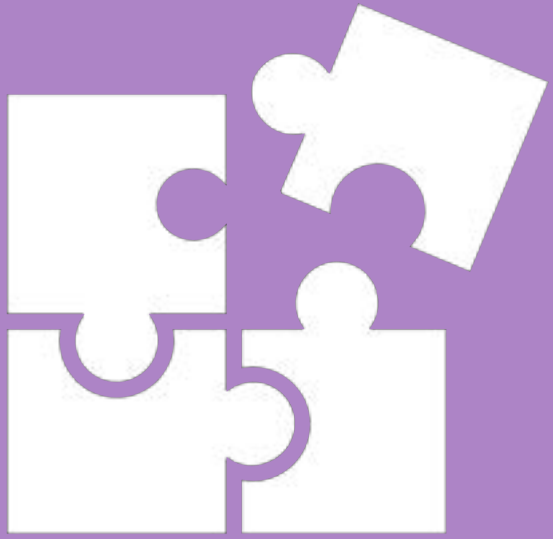
BROWSE · SIMPLE · ADVANCED | Pads | Warm Synth Pad | File · Save · Save As | Quality: Great | Limiter | Vol

Category	Subcategory	Genre	Timbre	Rating	Preset (3459 Presets)
All	All	All	All	★★★☆☆	Warm Synth Pad
Arpeggiated	Acoustic	Ambient	Simple	★★★☆☆	Warm Vintage Analog Pads
Bass	Analog	Downtempo	Complex	★★★☆☆	Warming Bell
Brass	Bass	Drum and Bass	Clean	★★★☆☆	Warp Core
Drums	Bell	Dubstep	Distorted	★★★☆☆	Warp Field Disruptor
Guitars	Bowed	Electronica	Bright	★★★☆☆	Warped Phase
Keys	Chaotic	Urban	Dark	★★★☆☆	Warped Strings
Leads	Choir	House	Thin	★★★☆☆	Wash And Spin
Mallets	Cinematic	Industrial	Phat	★★★☆☆	Water Comb
Organs	Classic	Pop	Nice	★★★☆☆	Water Creature Bass
Pads	Dance	Rock	Nasty	★★★☆☆	Water Drop Sticks
Rhythmic	Digital	Soundtrack	Organic	★★★☆☆	Water Gear
Sound Effects	Drone	Techno	Metallic	★★★☆☆	Water Guitar
Soundscapes	Drums	Trance	Smooth	★★★☆☆	Water Helicopter
Strings	Electric	Electro	Glitchy	★★★☆☆	Water Jug Clockwork
Synths	Electronic	Funk	Warm	★★★☆☆	Water Jug Drums
Vocals	Ensemble	Jazz	Cold	★★★☆☆	Water Jug Smack Bass
Woodwinds	Evolving	Orchestral	Noisy	★★★☆☆	Water Keys

Comments: Aftertouch: Vibrato, Mod Wheel: Gate Seq

User Tags | User Tags · Edit | Search

PERFORM | X1: Sync | X2: Noise Tune



Conclusion

Through this culmination I learned…

- ❖ Importance of Time Management
- ❖ Troubleshooting and thinking outside the box
- ❖ There’s nothing wrong with starting from scratch. It’s part of the process
- ❖ Programing/coding may look intimidating, and it’s okay.

Thank You