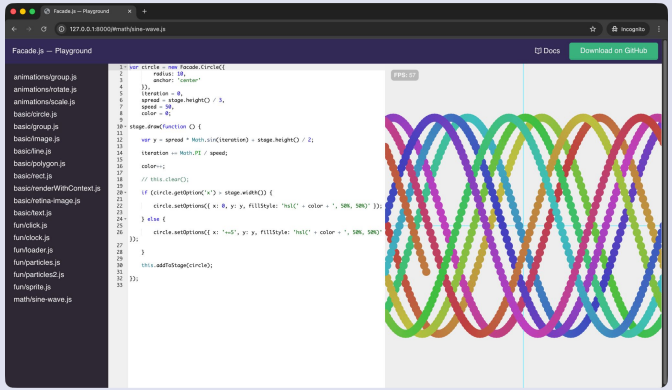
How to Build a Rhythm Game in (Almost) Any Game Engine

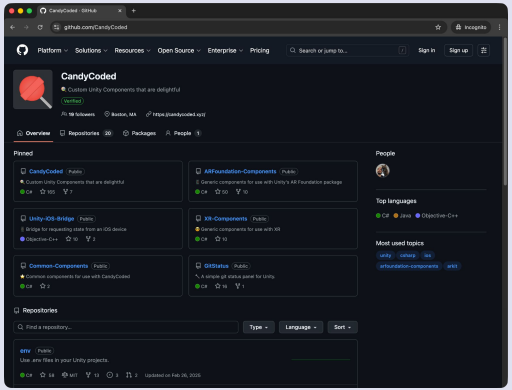
Boston Game Dev Week – MiniCon 2026

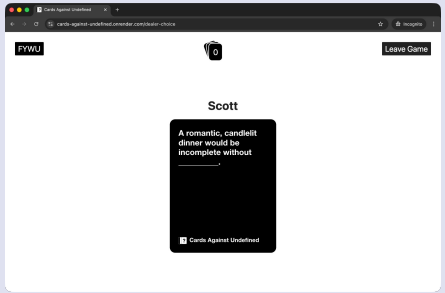
Who am I?

- Web / Game Developer
- Worked with Unity, Unreal, and Godot
- Built games as well as consumer based products in game engines
- Release a lot of what I do as open source projects
- I love data-in-data-out systems, writing unit tests and regular expressions (seriously)

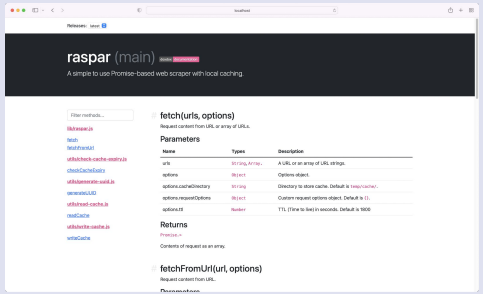
What Have I Built (Open Source Tools)



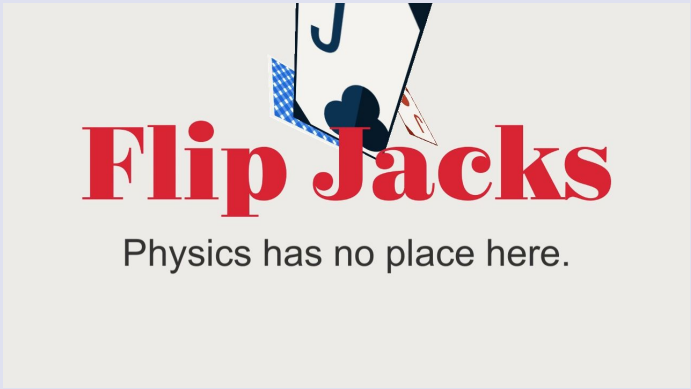


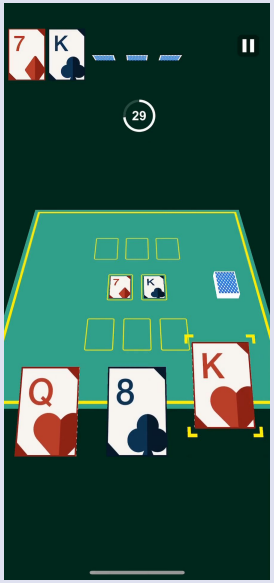


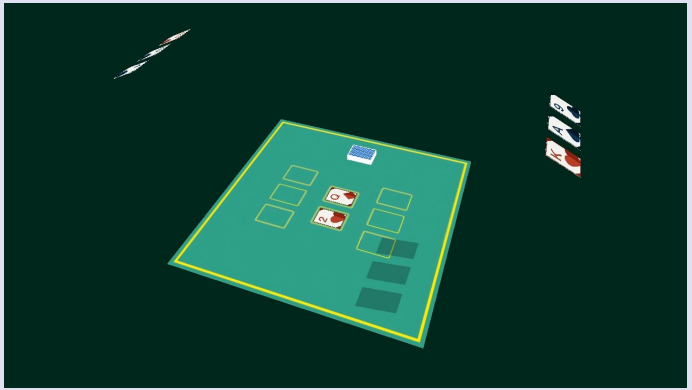
And many, many more tools.



What Have I Built (Games)







https://flipjacksgame.com/

My Favorite Rhythm Games

- PaRappa the Rapper (Playstation 1997)
- Guitar Hero (PS2 - 2005)
- Rock Band (XBOX - 2007)
- Patapon (PSP - 2008)
- DJ Hero (XBOX - 2009)
- Amplitude (PS4 - 2016)
- Audica (VR - 2019)
- Hi-Fi Rush (PC - 2023)

My Favorite Rhythm Games

- PaRappa the Rapper (Playstation 1997)
- Guitar Hero (PS2 - 2005)
- Rock Band (XBOX - 2007)
- Patapon (PSP - 2008)
- DJ Hero (XBOX - 2009)
- Amplitude (PS4 - 2016)
- Audica (VR - 2019)
- Hi-Fi Rush (PC - 2023)

Disclaimer:

I’m awful at rhythm games

Disclaimer:

I’m awful at rhythm games

& I’m not a musician. 😬

But I’ve talked to people who are:

1. **Good at rhythm games**
2. **Actual musicians**

https://rhythmgameutilities.com/

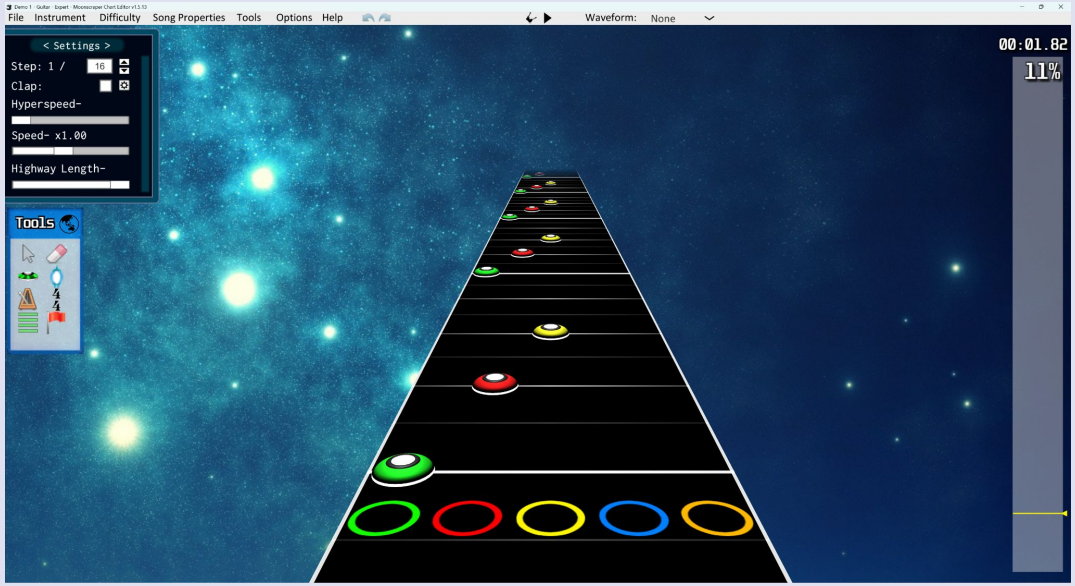
Let’s Break Down Rhythm Games

Rhythm Games Have Complicated Systems

1. Charting Notes
2. Loading Notes
3. Rendering Notes
4. Syncing Notes With Audio
5. Calculating Accuracy From User Input
6. Beat Matching
7. (Bonus) Input Devices (Midi/Custom)
8. (Bonus) Handling Input Latency

1. Charting Notes

1. Charting Notes (.chart)

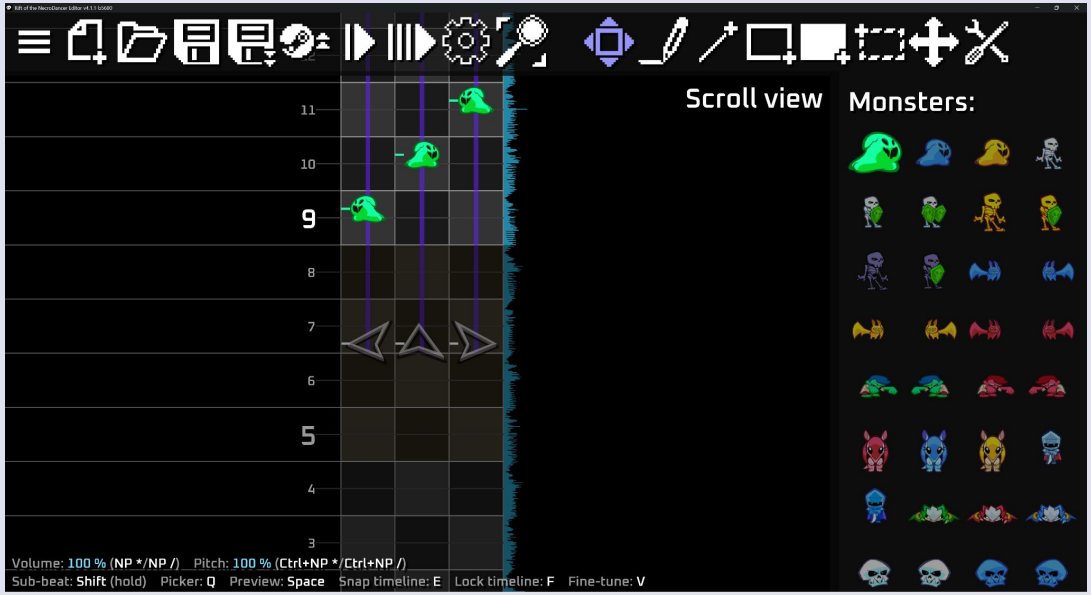


https://github.com/FireFox2000000/Moonscraper-Chart-Editor

1. Charting Notes (.mid)

https://signalmidi.app/

1. **Charting Notes** (In Game)



Rift of the NecroDancer Level Editor

2. Loading Notes

2. Loading Notes (.chart)

song.chart


```
[Song]
{
  Name = "Demo 1"
  Artist = "Scott Doxey"
  Charter = "Scott Doxey"
  Album = "Demo"
  Year = ", 2024"
  Offset = 0
  Resolution = 192
  Player2 = bass
  Difficulty = 0
  PreviewStart = 0
  PreviewEnd = 0
  Genre = "rock"
  MediaType = "cd"
  MusicStream = "song.ogg"
}
```

```
[SyncTrack]
{
  0 = TS 4
  0 = B 120000
}
```

```
[ExpertSingle]
{
  768 = N 0 0
  960 = N 1 0
  1152 = N 2 0
  1536 = N 0 0
  1728 = N 1 0
  1920 = N 2 0
  2304 = N 0 0
  2496 = N 1 0
  2688 = N 2 0
  3072 = N 0 0
  3264 = N 1 0
  3456 = N 2 0
  3840 = N 0 0
  4032 = N 1 0
  4224 = N 2 0
  4608 = N 0 0
  4800 = N 1 0
  4992 = N 2 0
  5376 = N 0 0
  5568 = N 1 0
  5760 = N 2 0
}
```

2. Loading Notes (.chart)

```
[SyncTrack]
{
  0 = TS 4
  0 = B 120000
}
```

```
[ExpertSingle]
{
  768 = N 0 0
  960 = N 1 0
  1152 = N 2 0
  1536 = N 0 0
  ...
}
```

First value is the tick (position)

2. Loading Notes (.chart)

```
[SyncTrack]
{
  0 = TS 4          TS = Time Signature
  0 = B 120000      B = BPM
}
```

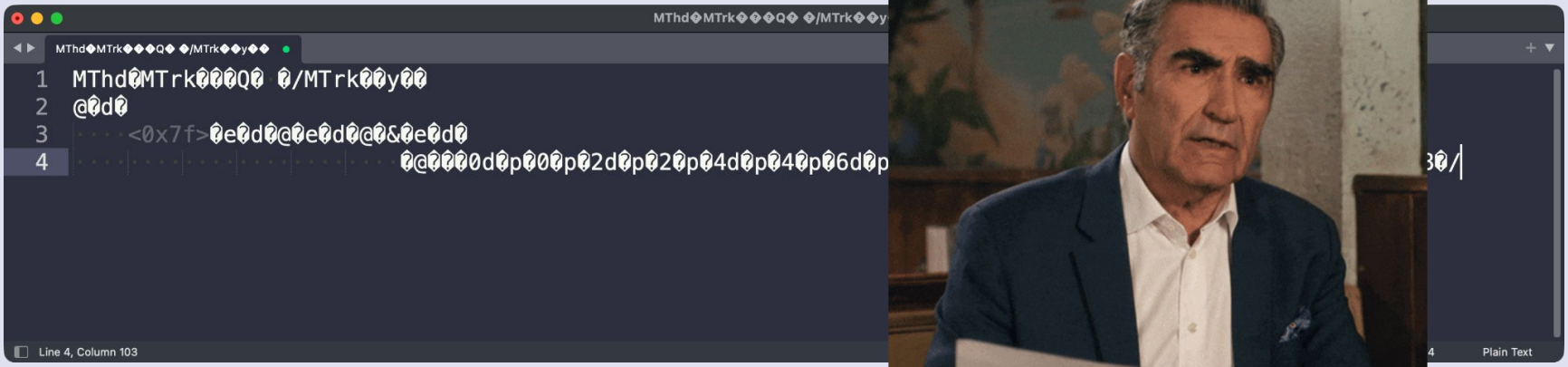
2. Loading Notes (.chart)

```
[ExpertSingle]
{
  768 = N 0 0
  960 = N 1 0
  1152 = N 2 0
  1536 = N 0 0
  ...
}
```

```
Note
Hand Position
Length (Hold Notes)
N 1 0
```

2. Loading Notes (.mid)

2. Loading Notes (.mid)



2. Loading Notes (.chart)

song.chart


```
[Song]
{
  Name = "Demo 1"
  Artist = "Scott Doxey"
  Charter = "Scott Doxey"
  Album = "Demo"
  Year = ", 2024"
  Offset = 0
  Resolution = 192
  Player2 = bass
  Difficulty = 0
  PreviewStart = 0
  PreviewEnd = 0
  Genre = "rock"
  MediaType = "cd"
  MusicStream = "song.ogg"
}
```

```
[SyncTrack]
{
  0 = TS 4
  0 = B 120000
}
```

```
[ExpertSingle]
{
  768 = N 0 0
  960 = N 1 0
  1152 = N 2 0
  1536 = N 0 0
  1728 = N 1 0
  1920 = N 2 0
  2304 = N 0 0
  2496 = N 1 0
  2688 = N 2 0
  3072 = N 0 0
  3264 = N 1 0
  3456 = N 2 0
  3840 = N 0 0
  4032 = N 1 0
  4224 = N 2 0
  4608 = N 0 0
  4800 = N 1 0
  4992 = N 2 0
  5376 = N 0 0
  5568 = N 1 0
  5760 = N 2 0
}
```

2. Loading Notes (.chart)

```
inline std::regex CHART_SECTION_PATTERN(R"(\[([a-z]+)\]\s*\{([^\}]+)\})",
                                        std::regex_constants::icase);

inline std::regex CHART_SECTION_LINE_PATTERN(R"(([^=]+)\s*=([^\r\n]+))");

inline std::regex JSON_VALUE_PATTERN(R"(("[^"]+"|\S+))");
```

Yes, I actually wrote these by hand because I'm nuts.

2. Loading Notes

- Song.FromChartData
- Song.FromMidiData
- ReadNotesFromChartData
- ReadNotesFromMidiData

```
npx chart-to-json ./notes.chart > notes.json
```

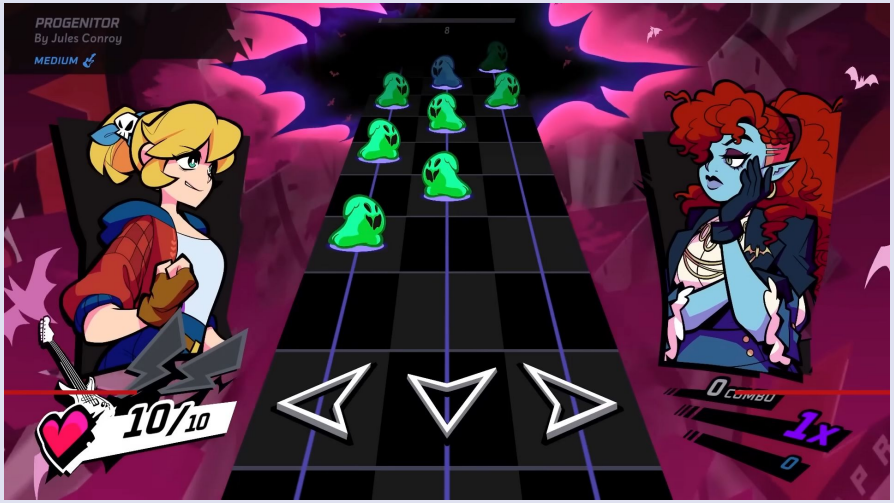

```
{
  "Name": "Demo 1",
  "Artist": "Scott Doxey",
  "Album": "Demo",
  "Genre": "rock",
  "Year": ", 2024",
  "Charter": "Scott Doxey",
  "Resolution": 192,
  "Difficulty": 0,
  "Offset": 0,
  "PreviewStart": 0,
  "PreviewEnd": 0,
  "MusicStream": "song.ogg",
  "Lyrics": {},
  ...
}
```

```
{
  ...,
  "BPM": {
    "0": 120000
  },
  "TimeSignatures": {
    "0": [
      4
    ]
  }
}
```

```
{
  ...,
  "Difficulties": {
    ...,
    "Expert": [
      {
        "Position": 768,
        "HandPosition": 0,
        "Length": 0
      },
      {
        "Position": 960,
        "HandPosition": 1,
        "Length": 0
      },
      {
        "Position": 1152,
        "HandPosition": 2,
        "Length": 0
      },
      {
        "Position": 1536,
        "HandPosition": 0,
        "Length": 0
      },
      ...
    ]
  },
  ...
}
```


3. Rendering Notes

3. Rendering Notes

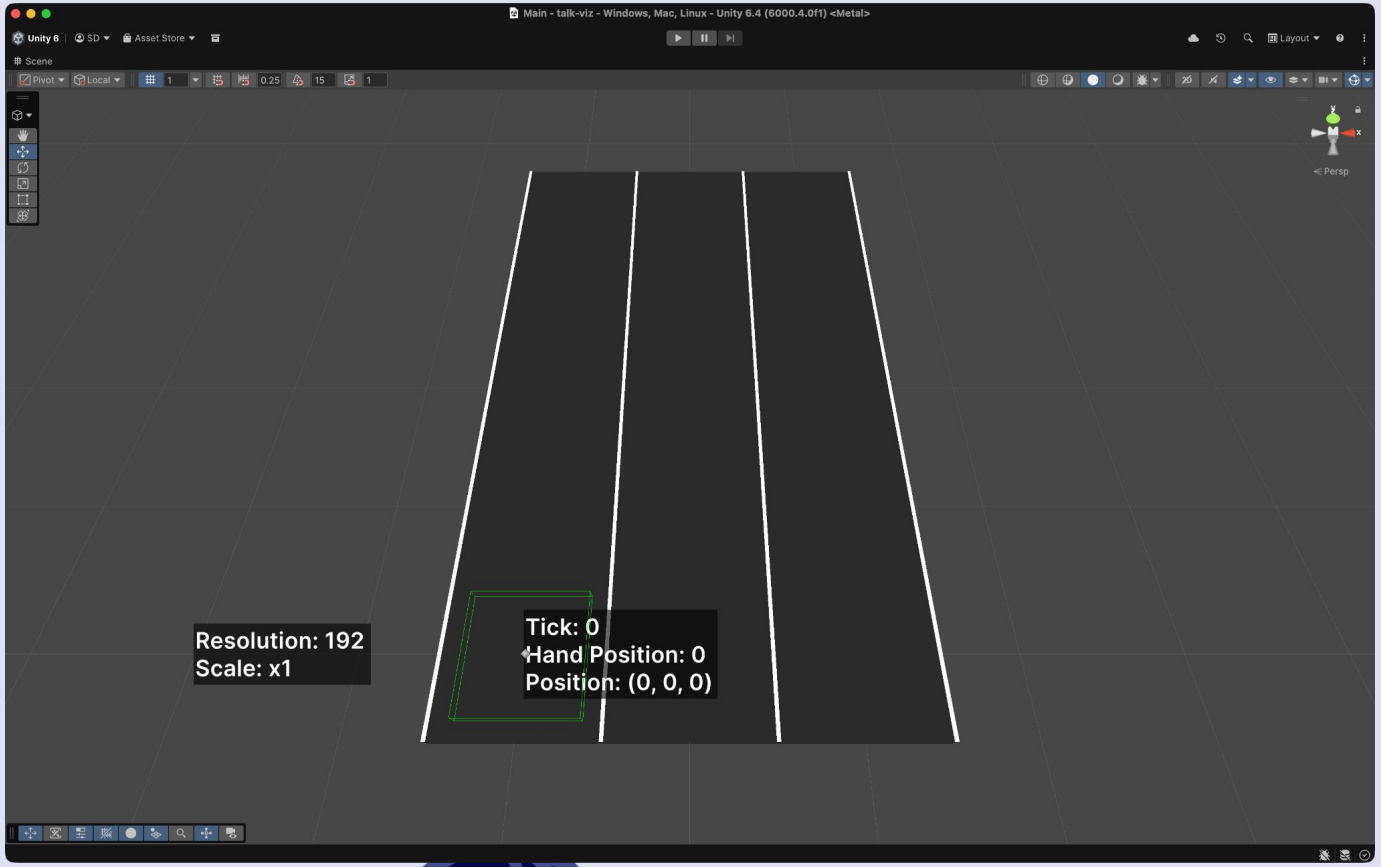


Rift of the NecroDancer

Amplitude (PS4)

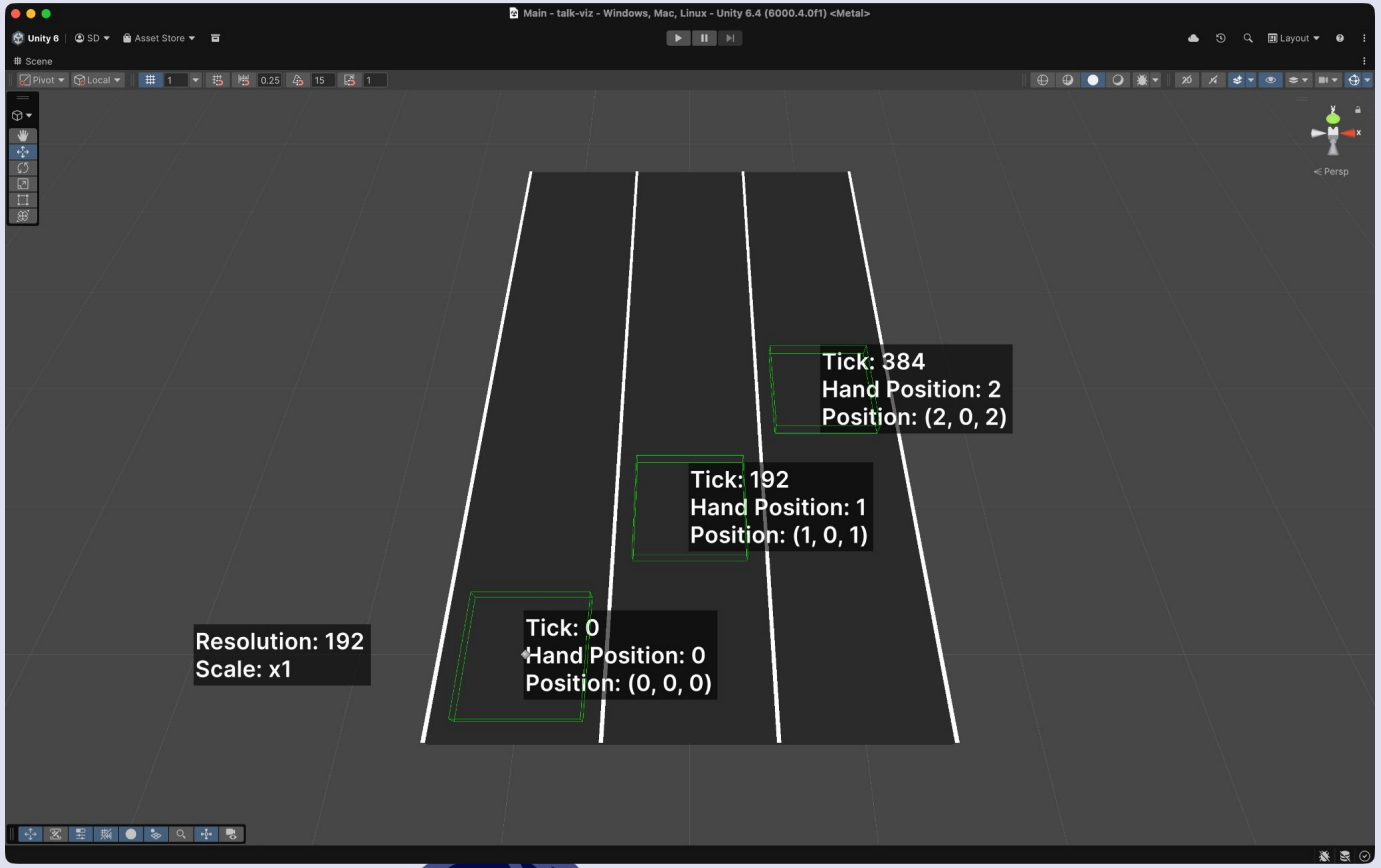
3. Rendering Notes

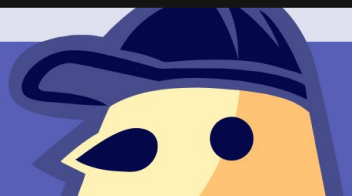
```
using System;
using RhythmGameUtilities;

const int tick = 0;
const int resolution = 192;

var position = Utilities.ConvertTickToPosition(tick, resolution);

Console.WriteLine(position); // 0
```


Main - talk-viz - Windows, Mac, Linux - Unity 6.4 (6000.4.0f1) <Metal>
Unity 6
Asset Store
Scene
Persp
Resolution: 192
Scale: x1
Tick: 0
Hand Position: 0
Position: (0, 0, 0)

3. Rendering Notes

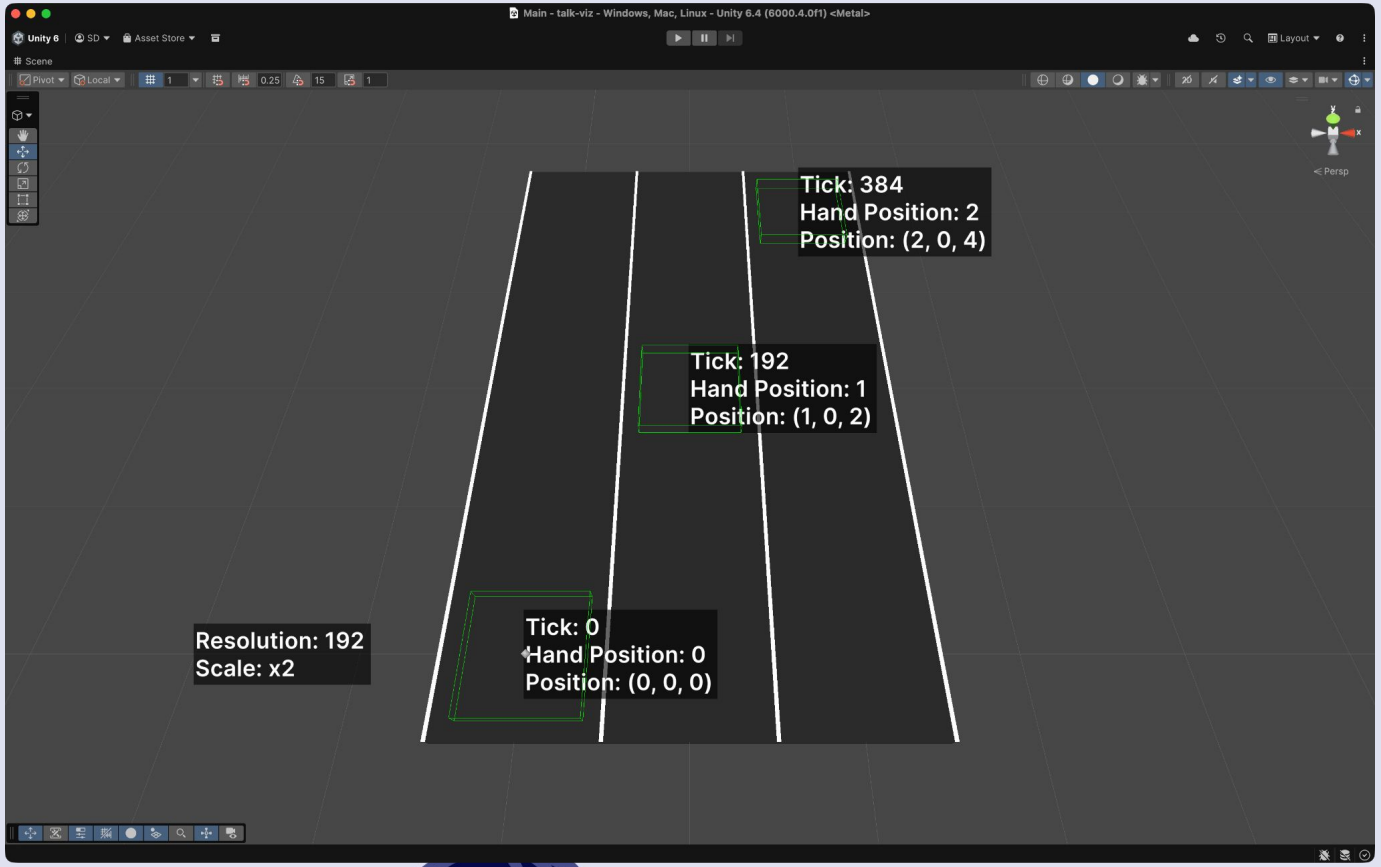
```
1  using System;
2  using RhythmGameUtilities;
3
4  const int resolution = 192;
5
6  Console.WriteLine(Utilities.ConvertTickToPosition(192, resolution)); // 1
7  Console.WriteLine(Utilities.ConvertTickToPosition(384, resolution)); // 2
```


Main - talk-viz - Windows, Mac, Linux - Unity 6.4 (6000.4.0f1) <Metal>
Unity 6
Asset Store
Scene
Persp
Tick: 384
Hand Position: 2
Position: (2, 0, 2)
Tick: 192
Hand Position: 1
Position: (1, 0, 1)
Tick: 0
Hand Position: 0
Position: (0, 0, 0)
Resolution: 192
Scale: x1



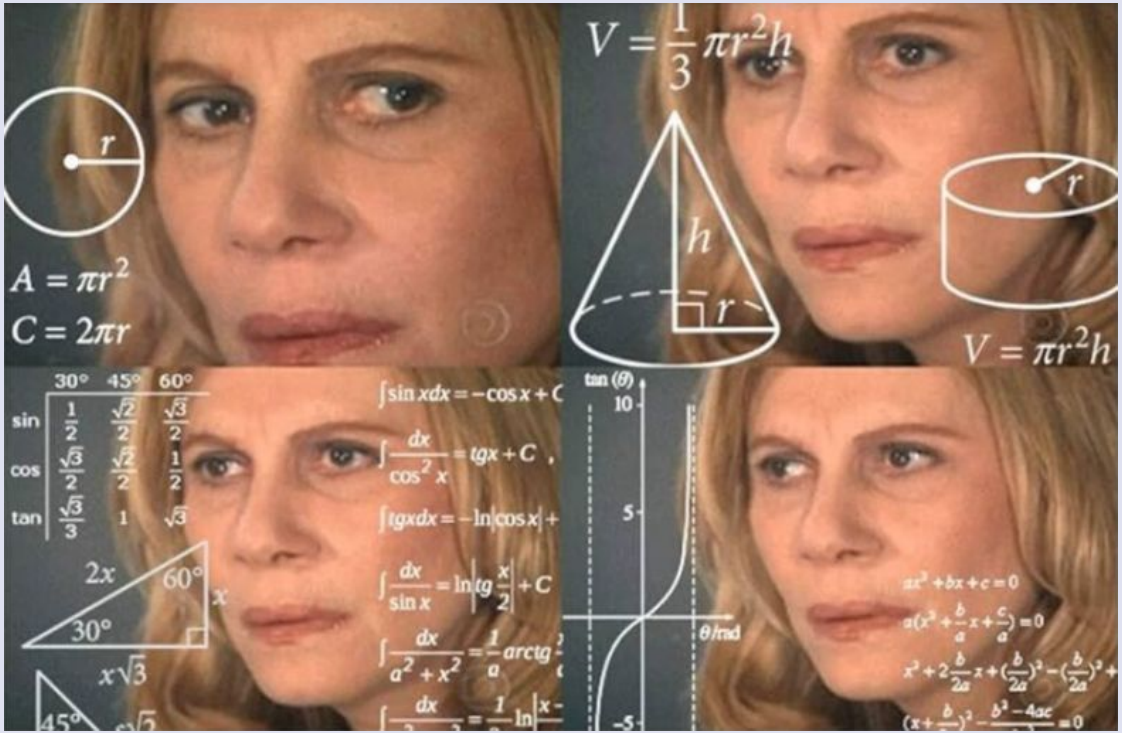
3. Rendering Notes

```
using System;
using RhythmGameUtilities;

const int tick = 384;
const int resolution = 192;
const int scale = 2

var position = Utilities.ConvertTickToPosition(tick, resolution) * scale;

Console.WriteLine(position); // 4
```


Main - talk-viz - Windows, Mac, Linux - Unity 6.4 (6000.4.0f1) <Metal>
Tick: 384
Hand Position: 2
Position: (2, 0, 4)
Tick: 192
Hand Position: 1
Position: (1, 0, 2)
Tick: 0
Hand Position: 0
Position: (0, 0, 0)
Resolution: 192
Scale: x2

4. Syncing Notes With Audio

So this section gets a bit …


$A = \pi r^2$
$C = 2\pi r$
$V = \frac{1}{3}\pi r^2 h$
$V = \pi r^2 h$
$\int \sin x dx = -\cos x + C$
$\int \frac{dx}{\cos^2 x} = tgx + C$
$\int tgx dx = -\ln|\cos x| +$
$\int \frac{dx}{\sin x} = \ln\left|tg\frac{x}{2}\right| + C$
$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} arctg$
$\tan(\theta)$
θ/rad
$ax^2 + bx + c = 0$

Disclaimer:

I’m awful at rhythm games

& I’m not a musician. 😬

Disclaimer:

I’m awful at rhythm games

& I’m not a musician. 😬

(and I’m not great at math)

But I’ve talked to people who are:

1. Good at rhythm games

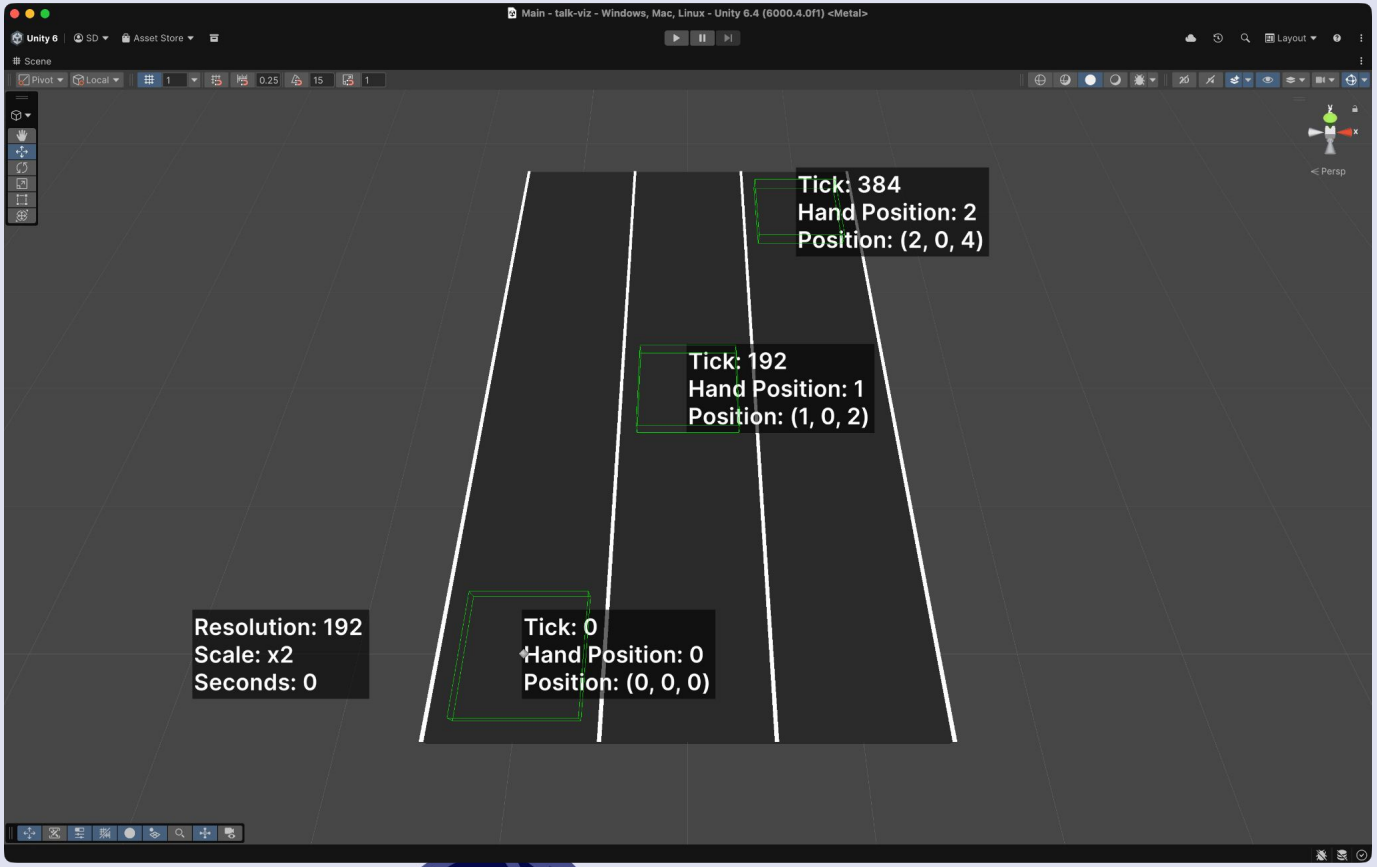
2. Actual musicians

But I’ve talked to people who are:

1. **Good at rhythm games**
2. **Actual musicians**
3. **Good at math**

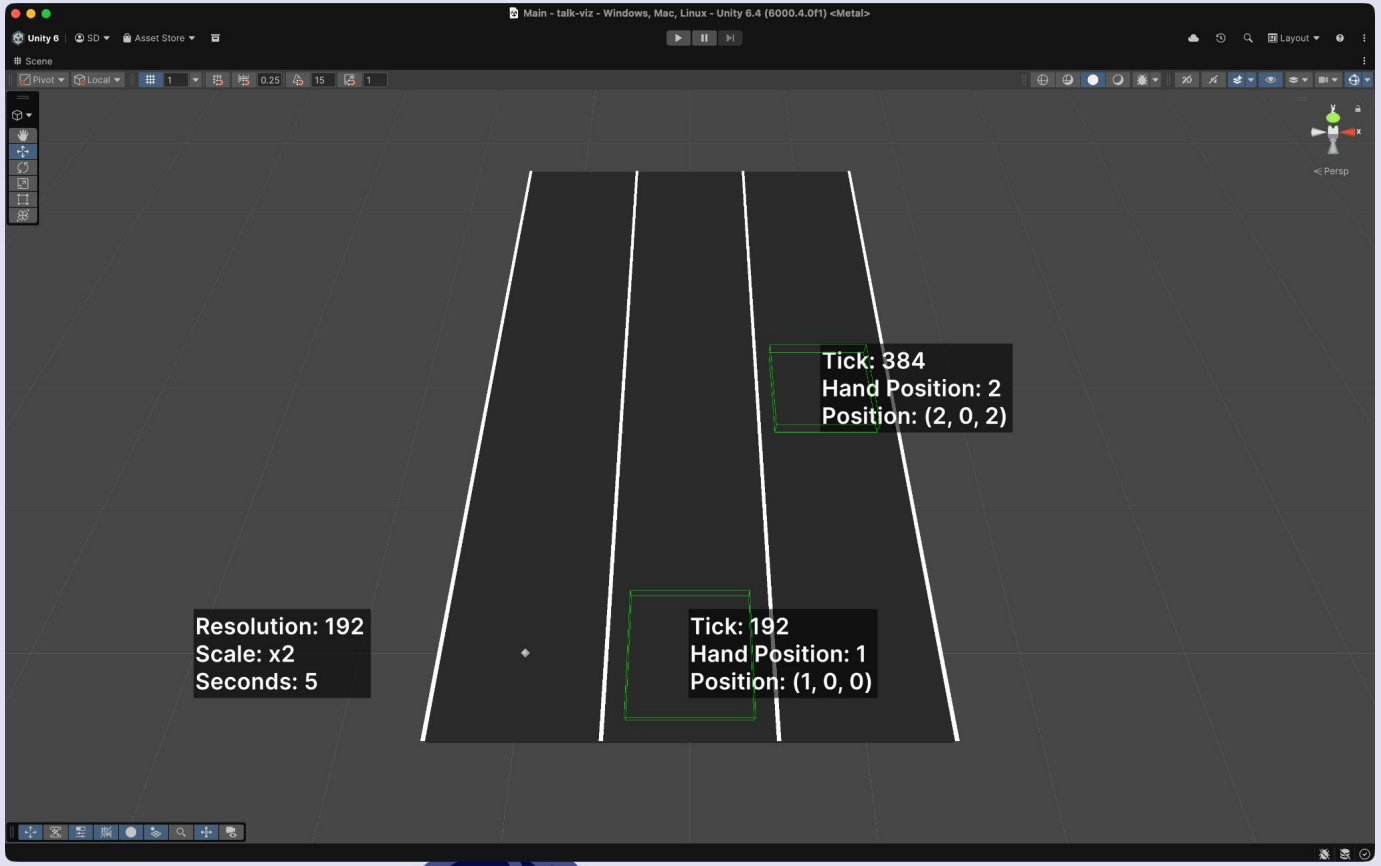
4. Syncing Notes With Audio

```
1  var currentTime = 0;
2
3  var resolution = 192;
4  var tempoChanges = new Tempo[] { new() { Position = 0, BPM = 12000 } };
5
6  var tickOffset = Utilities.ConvertSecondsToTicks(currentTime, resolution, tempoChanges);
7
8  Console.WriteLine(tickOffset); // 0
9
10 var note = new Note { Position = 0, HandPosition = 0, Length = 0 };
11
12 var position = Utilities.ConvertTickToPosition(note.Position - tickOffset, resolution);
13
14 Console.WriteLine(position); // 0
```


Main - talk-viz - Windows, Mac, Linux - Unity 6.4 (6000.4.0f1) <Metal>
Tick: 384
Hand Position: 2
Position: (2, 0, 4)
Tick: 192
Hand Position: 1
Position: (1, 0, 2)
Tick: 0
Hand Position: 0
Position: (0, 0, 0)
Resolution: 192
Scale: x2
Seconds: 0

4. Syncing Notes With Audio

```
1  var currentTime = 5;
2
3  var resolution = 192;
4  var tempoChanges = new Tempo[] { new() { Position = 0, BPM = 12000 } };
5
6  var tickOffset = Utilities.ConvertSecondsToTicks(currentTime, resolution, tempoChanges);
7
8  Console.WriteLine(tickOffset); // 192
9
10 var note = new Note { Position = 0, HandPosition = 0, Length = 0 };
11
12 var position = Utilities.ConvertTickToPosition(note.Position - tickOffset, resolution);
13
14 Console.WriteLine(position); // -1
```


Main - talk-viz - Windows, Mac, Linux - Unity 6.4 (6000.4.0f1) <Metal>
Tick: 384
Hand Position: 2
Position: (2, 0, 2)
Resolution: 192
Scale: x2
Seconds: 5
Tick: 192
Hand Position: 1
Position: (1, 0, 0)

4. Syncing Notes With Audio

```
var currentTime = 5;

var resolution = 192;
var tempoChanges = new Tempo[]
{
    new() { Position = 0, BPM = 88000 }, new() { Position = 3840, BPM = 112000 },
    new() { Position = 9984, BPM = 89600 }, new() { Position = 22272, BPM = 112000 },
    new() { Position = 33792, BPM = 111500 }, new() { Position = 34560, BPM = 112000 },
    new() { Position = 42240, BPM = 111980 }
};

var tickOffset = Utilities.ConvertSecondsToTicks(currentTime, resolution, tempoChanges);

Console.WriteLine(tickOffset); // 1408

var note = new Note { Position = 0, HandPosition = 0, Length = 0 };

var position = Utilities.ConvertTickToPosition(note.Position - tickOffset, resolution);

Console.WriteLine(position); // -7.3333335
```

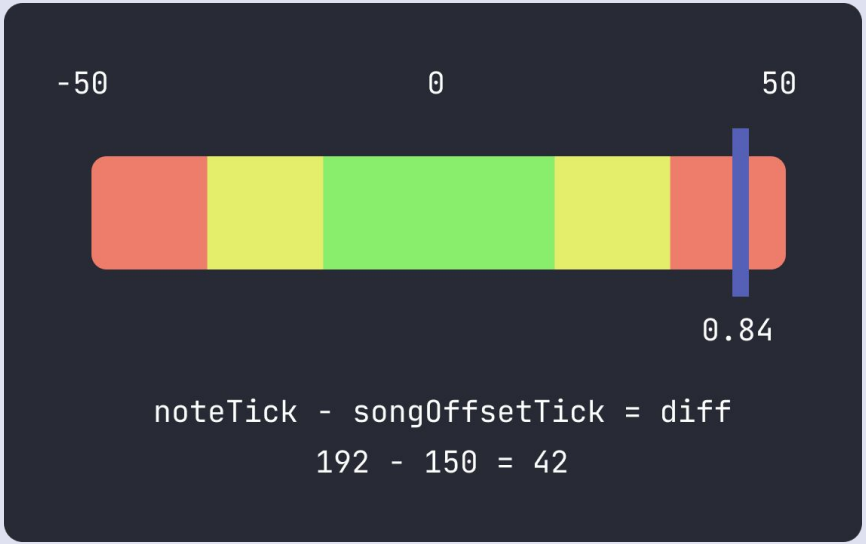

No thanks

5. Calculating Accuracy From User Input

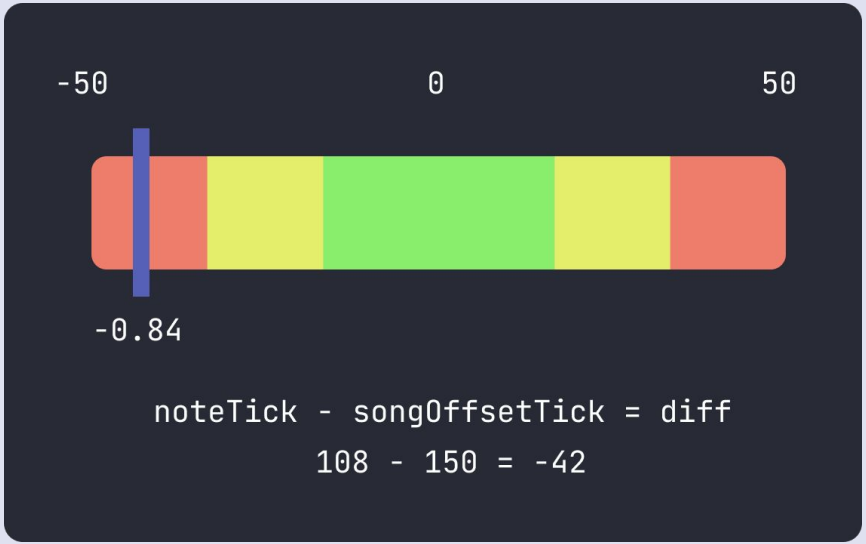
5. Calculating Accuracy of Hit Notes From User Input

```
var currentPosition = Utilities.ConvertSecondsToTicks(
  currentTime, resolution, tempoChanges);

var accuracyRatio = Utilities.CalculateAccuracyRatio(
  note.Position, currentPosition, delta);

Console.WriteLine(accuracy); // 0.64
```

5. Calculating Accuracy of Hit Notes From User Input



5. Calculating Accuracy of Hit Notes From User Input



5. Calculating Accuracy of Hit Notes From User Input

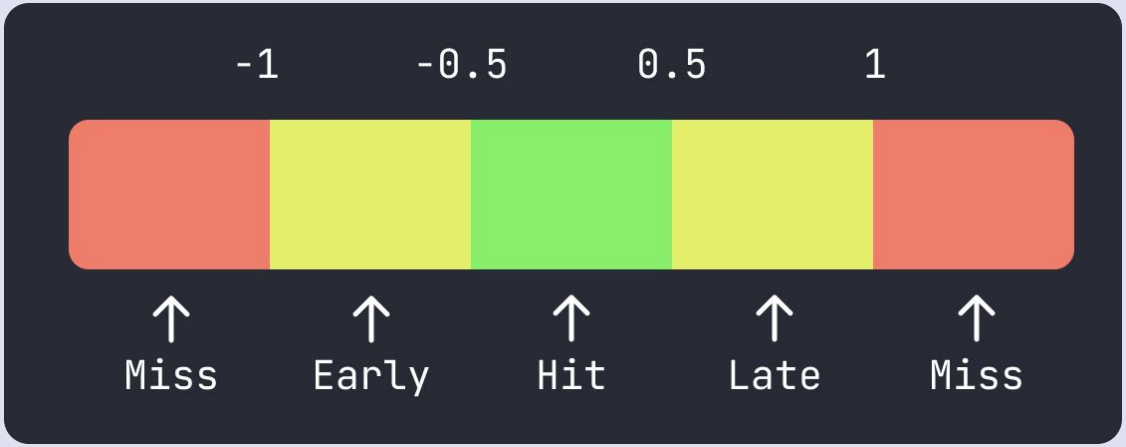
```
1  var currentPosition = Utilities.ConvertSecondsToTicks(
2    currentTime, resolution, tempoChanges);
3
4  var accuracy = Utilities.CalculateAccuracy(note.Position,
5    currentPosition, delta);
6
7  Console.WriteLine(accuracy); // Invalid, Poor, Fair, Good, Great, Perfect
```

5. Calculating Accuracy of Hit Notes From User Input

5. Calculating Accuracy of Hit Notes From User Input

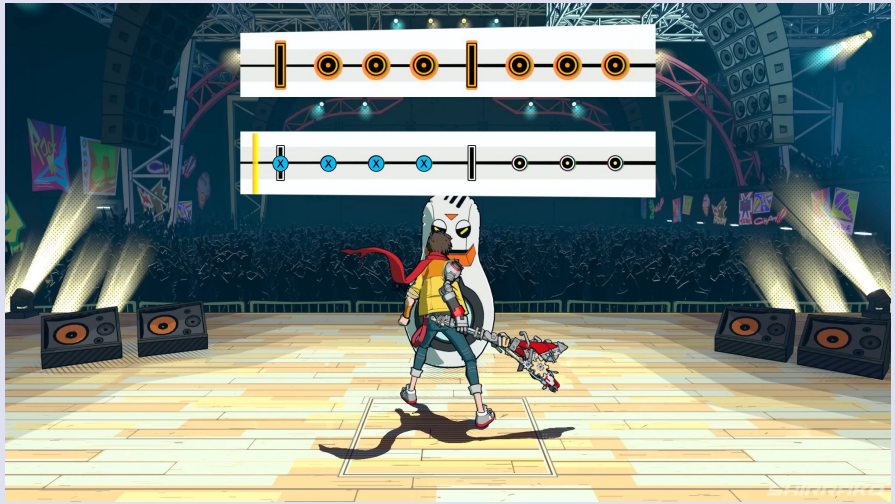
```
1  var currentPosition = Utilities.ConvertSecondsToTicks(
2    currentTime, resolution, tempoChanges);
3
4  var timing = Utilities.CalculateTiming(
5    note.Position, currentPosition, delta);
6
7  Console.WriteLine(timing); // Miss, Hit, Early, Late
```

5. Calculating Accuracy of Hit Notes From User Input



6. Beat Matching

6. Beat Matching



Hi-Fi Rush

Crypt of the NecroDancer

6. Beat Matching

```
1  using System;
2  using RhythmGameUtilities;
3
4  const int bpm = 120;
5  const float currentTime = 10f;
6  const float delta = 0.05f;
7
8  var isOnTheBeat = Utilities.IsOnTheBeat(bpm, currentTime, delta);
9
10 Console.WriteLine(isOnTheBeat); // true
```

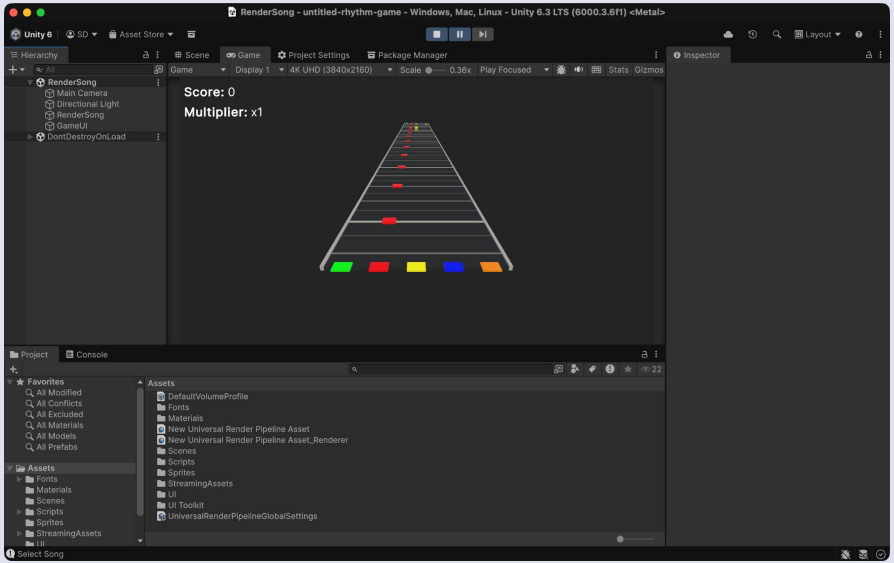
6. Beat Matching

```
1  private void Update()
2  {
3      elapsedTime += Time.deltaTime;
4
5      var isOnTheBeat = Utilities.IsOnTheBeat(120, elapsedTime);
6
7      if (!isOnBeatTriggered && isOnTheBeat)
8      {
9          Debug.Log("On Beat");
10
11         isOnBeatTriggered = true;
12     }
13     else if (isOnBeatTriggered && !isOnTheBeat)
14     {
15         isOnBeatTriggered = false;
16     }
17 }
```

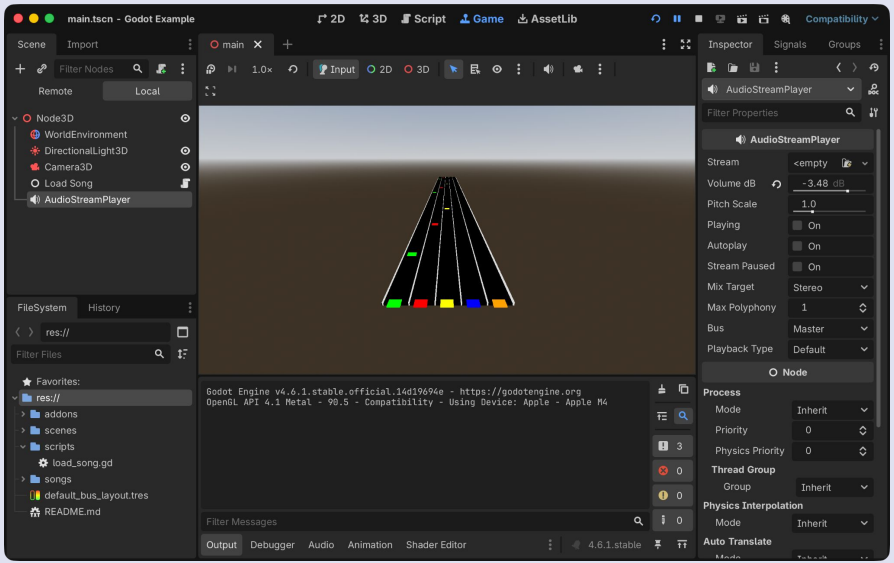
Upcoming Features

- Stepmania Import
- Love2D (Lua) Plugin
- Bevy (Rust) Plugin
- GameMaker (GML) Plugin
- Desktop/Web Charting Tool

Examples



Unity



Godot

Examples

SDL

Questions?

Thank you!