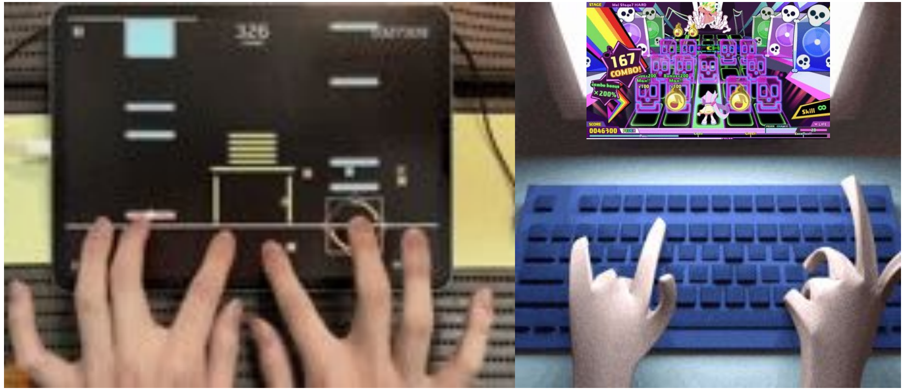
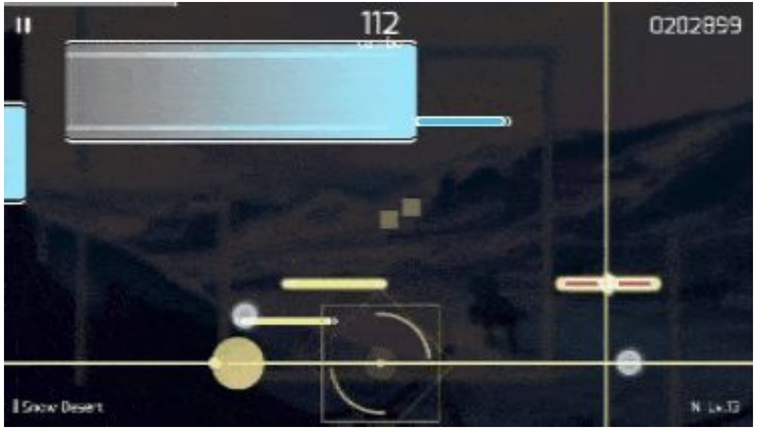
Team D6: Rhythm Genesis

– Yuhe Ma, Michelle Bryson, Lucas Storm



mobile vs. pc rhythm games



A sample beatmap from Phigros

Use Case

Popular rhythm games have two limitations:

1) Expensive

2) Not very customizable ✗

An engaging beatmap is hard to make!

Rhythm Genesis for Rhythm Game Players:

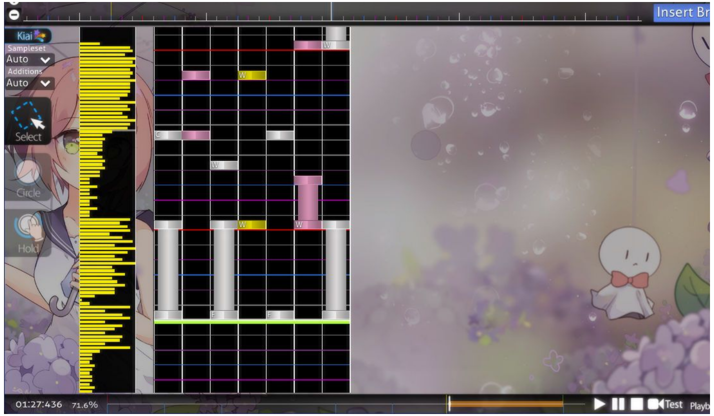
1) Free (available on Steam)

2) Fully customizable (auto-beatmap generation + in game beatmap editor)

Our project encompass areas in Software Engineering and Signal Processing

Requirements (high-level)

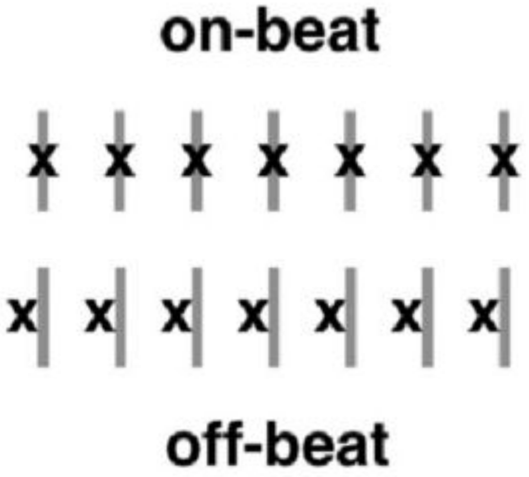
- Players can upload music files in supported formats (MP3, WAV, OGG, etc.)
- Players can create or auto generate beatmaps using their own music
- Responsive gameplay and precise timing and scoring for note hits



Osu! beatmap editor

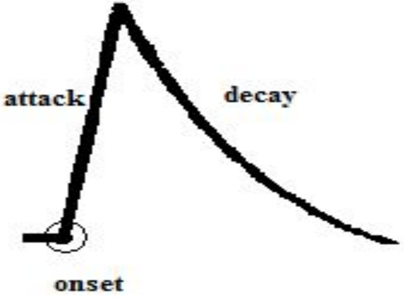
Requirements (gameplay)

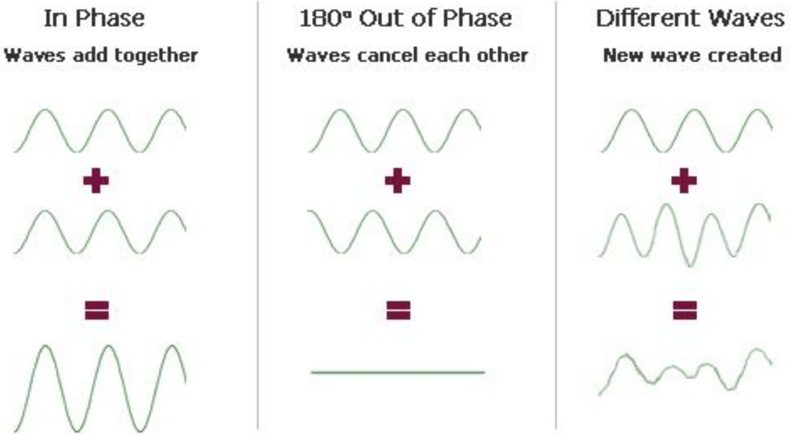
- Auto generation algorithm accuracy
  - Beat alignment error ≤ 20 ms
- Tempo detection range
  - 50-220 BPM (beats per minute)
- Performance
  - Minimum 30 FPS during gameplay
  - Song load and processing time ≤ 5 sec
- In-game beat map editor:
  - Saving a beat map ≤ 5 seconds



Technical Challenges (part 1)

- Accurate tempo & beat detection (Signal processing challenges)
- Testing Auto-Generated Beat Maps





Technical Challenges (part 2)

1. Synchronization of Falling Notes with the Beatmap
2. Performance Bottlenecks (FPS drops, long load & processing time)
3. Implementing a User-Friendly Beatmap Editor
4. Real-Time Scoring & Feedback System
5. Publishing on Steam

Solution Approach (high-level)

Development Stack:

🎮 **Unity (C#)** – Core gameplay, UI, animations, physics

🎵 **Librosa (Python)** – Audio analysis for tempo, beats & onset detection

Workflow:

1️⃣ **Core Game Loop (Unity)** → UI, gameplay, real-time sync, timing accuracy, feedback system, hardcoded beatmaps

2️⃣ **Audio Processing (Librosa)** → Detects beats & generates beatmaps in JSON

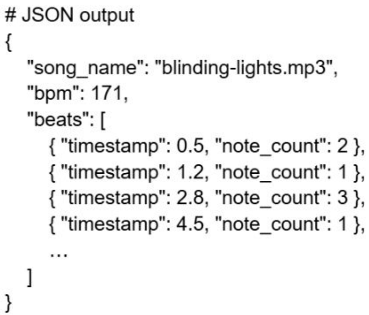
3️⃣ **Beatmap Integration (Unity)** → Reads JSON to spawn notes

4️⃣ **In-Game Beatmap Editor (Unity)** → customizable beatmap editor with waveform visualization

5️⃣ **Performance & Optimization (Unity)** → Async processing, object pooling, advanced UI & visual effects, 60 FPS target

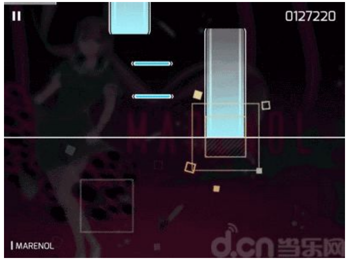
6️⃣ **Publishing (Unity & Steam)** → Steam integration, cross-platform compatibility

Librosa



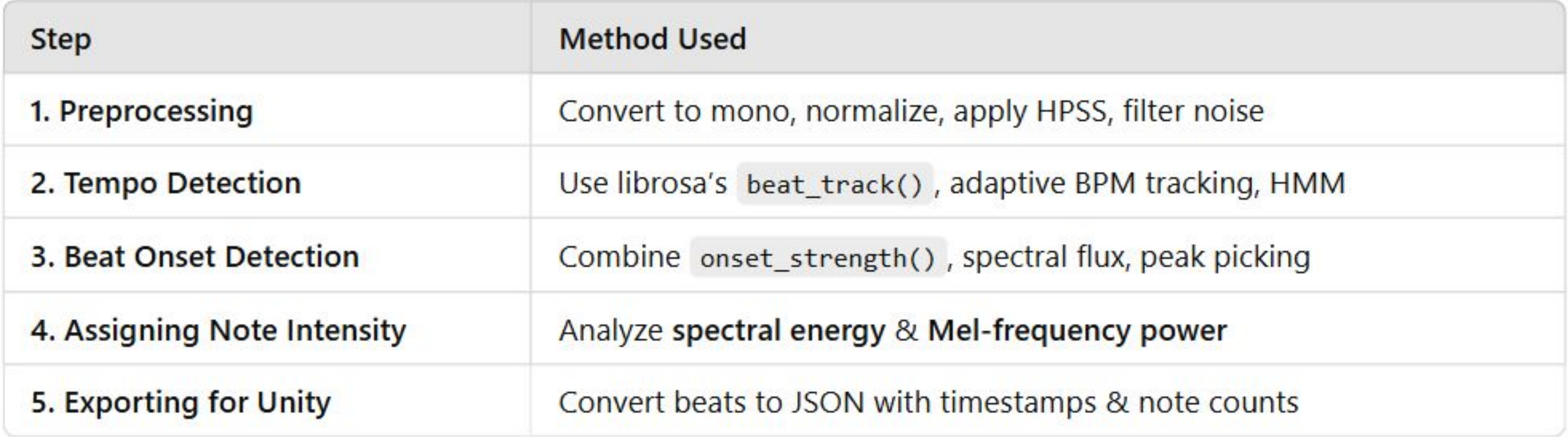
```
# JSON output
{
   "song_name": "blinding-lights.mp3",
   "bpm": 171,
   "beats": [
      { "timestamp": 0.5, "note_count": 2 },
      { "timestamp": 1.2, "note_count": 1 },
      { "timestamp": 2.8, "note_count": 3 },
      { "timestamp": 4.5, "note_count": 1 },
      ...
   ]
}
```

Unity



Solution Approach (audio processing)

Detecting **beats and tempo** accurately from any audio file is a **challenging signal processing problem** due to **diverse musical structures**.



Step	Method Used
1. Preprocessing	Convert to mono, normalize, apply HPSS, filter noise
2. Tempo Detection	Use librosa's `beat_track()` , adaptive BPM tracking, HMM
3. Beat Onset Detection	Combine `onset_strength()` , spectral flux, peak picking
4. Assigning Note Intensity	Analyze **spectral energy** & **Mel-frequency power**
5. Exporting for Unity	Convert beats to JSON with timestamps & note counts

Solution Approach (misc.)

- Synchronization of Falling Notes (Game Objects) with the Beatmap
  - Adjust note spawn timing based on Unity’s audio DSP time
- Performance Bottlenecks (FPS drops, long load & processing time)
  - Object Pooling
- Implementing a User-Friendly Beatmap Editor: Grid-based Snapping System
- Real-Time Scoring & Feedback System
  - Perfect (≤10ms), Good (≤20ms), Bad (≤40ms), Miss (>40ms).
- Publishing on Steam & Cross Platform Compatibility: Unity Documentation

Testing, Verification, Metrics

Beatmaps: beat alignment error < 20ms 90% of the time

- Use manually created MIDI files (so we know exact timing of each note)
- Test songs with BPM between 30 and 250

Gameplay Responsiveness:

- Use many human testers > 20 and ask for their feedback
- Steam User Feedback

Performance:

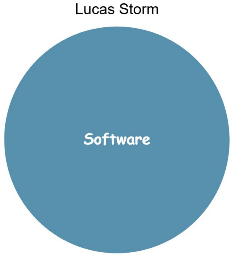
- Stress testing with large music files to ensure load/processing time < 5s

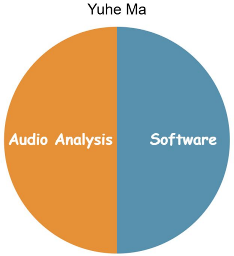
✅ Automate Testing Where Possible: Use Python scripts to measure beat detection accuracy and BPM deviation.

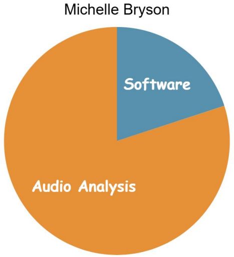
✅ Profile in Unity: Use Unity's Performance Profiler to analyze FPS bottlenecks.

Tasks/division of labor

1. **Core Game Loop (Unity)** → Yuhe Ma & Lucas Storm
2. **Audio Processing (Librosa)** → Michelle Bryson & Yuhe Ma
3. **Beatmap Integration (Unity)** → Yuhe Ma & Lucas Storm
4. **In-Game Beatmap Editor (Unity)** → Lucas Storm & Yuhe Ma
5. **Performance & Optimization (Unity)** → Lucas Storm & Yuhe Ma
6. **Advanced UI & Visual Effects (Unity)** → Michelle Bryson
7. **Testing & Publishing (Unity & Steam)** → ALL OF US

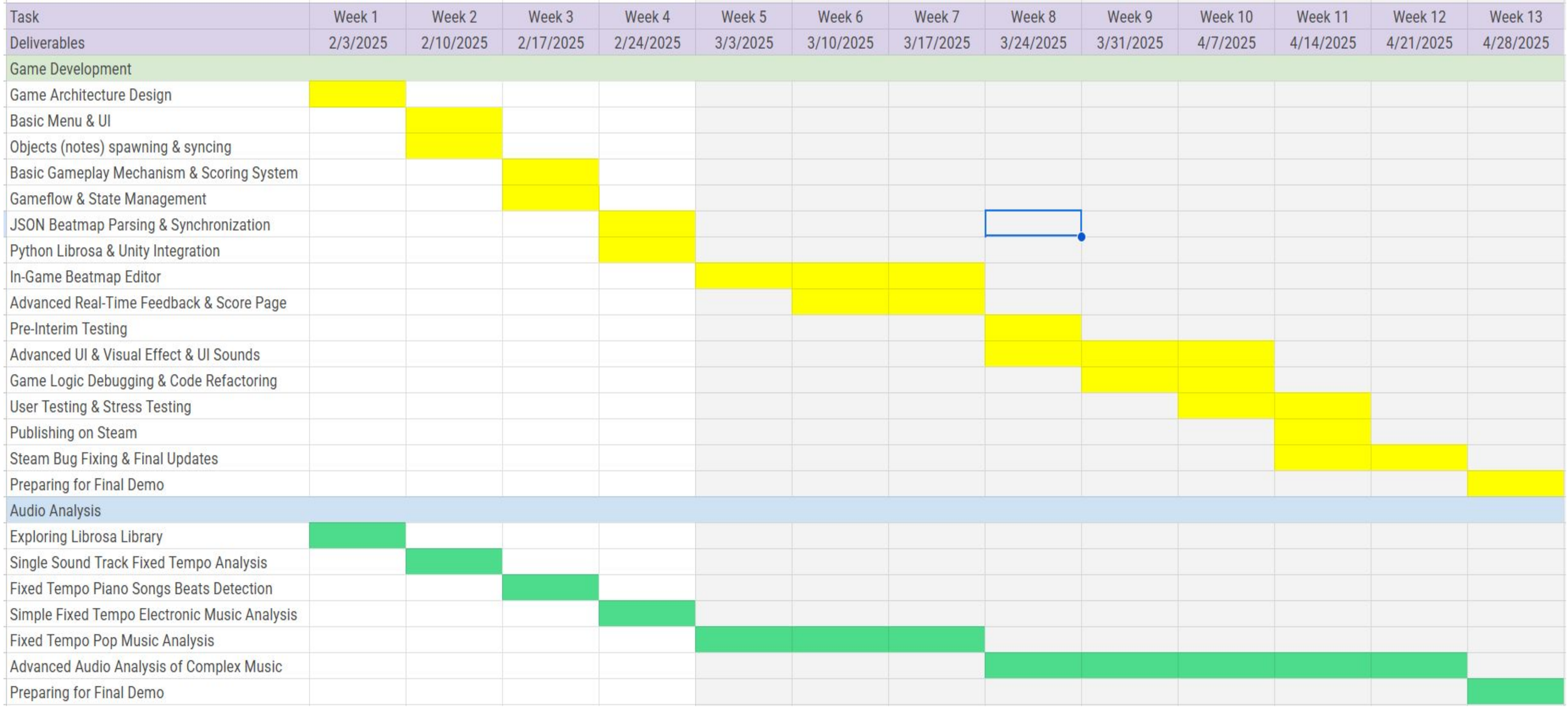






Schedule - Gantt Chart

Rush E Gamplay (A Dance of Fire and Ice):
https://youtu.be/r5PV14QKLN8?si=kMIrrfu-e4k1v8kO



Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Deliverables	2/3/2025	2/10/2025	2/17/2025	2/24/2025	3/3/2025	3/10/2025	3/17/2025	3/24/2025	3/31/2025	4/7/2025	4/14/2025	4/21/2025	4/28/2025
Game Development													
Game Architecture Design	🟨												
Basic Menu & UI		🟨											
Objects (notes) spawning & syncing		🟨											
Basic Gameplay Mechanism & Scoring System			🟨										
Gameflow & State Management			🟨										
JSON Beatmap Parsing & Synchronization				🟨									
Python Librosa & Unity Integration				🟨									
In-Game Beatmap Editor					🟨	🟨	🟨						
Advanced Real-Time Feedback & Score Page						🟨	🟨						
Pre-Interim Testing								🟨					
Advanced UI & Visual Effect & UI Sounds								🟨	🟨	🟨			
Game Logic Debugging & Code Refactoring									🟨	🟨			
User Testing & Stress Testing										🟨	🟨		
Publishing on Steam											🟨		
Steam Bug Fixing & Final Updates											🟨	🟨	
Preparing for Final Demo													🟨
Audio Analysis													
Exploring Librosa Library	🟩												
Single Sound Track Fixed Tempo Analysis		🟩											
Fixed Tempo Piano Songs Beats Detection			🟩										
Simple Fixed Tempo Electronic Music Analysis				🟩									
Fixed Tempo Pop Music Analysis					🟩	🟩	🟩						
Advanced Audio Analysis of Complex Music								🟩	🟩	🟩	🟩	🟩	
Preparing for Final Demo													🟩