ingenieur
wissenschaften
htw saar

Game Design and Development

Wave Voyager – Prototype 2

Chris Loch, Jan-Eric Schober

Table of contents

- Overall status
  - Tasks finished
  - Tasks open
- Development
- Beat Detection Algorithm
  - Architecture idea
  - Challenges
  - Actual functionality
- Visual Outcome
- Demo
- Questions

Overall status

Tasks finished:

- (Kind of) Working beat detection algorithm.
- Basic player movement.
- Obstacle spawning.
- Menu layout.

Tasks open:

- Effects and further visualizations.
- Track to ride along.
- Useful HUD.
- Tweaking and more precise beat detection.

Table of contents

- Overall status
  - Tasks finished
  - Tasks open
- Development
- Beat Detection Algorithm
  - Architecture idea
  - Challenges
  - Actual functionality
- Visual Outcome
- Demo
- Questions

Development

Methodology

- Weekly meetups (Task/sprint coordination, discussions about problems).
- Individual sprints for objectives.
- The less spaghetti, the better.

Tools

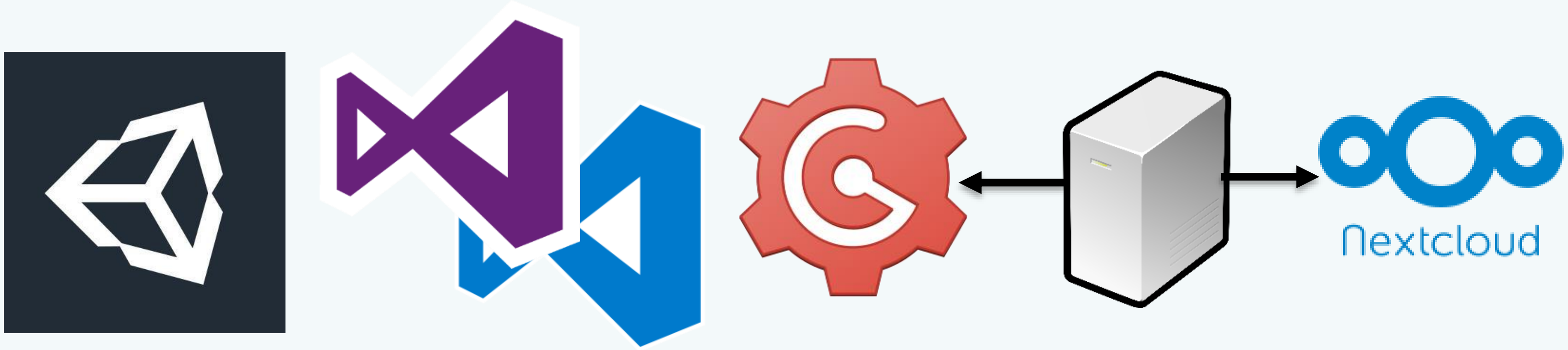
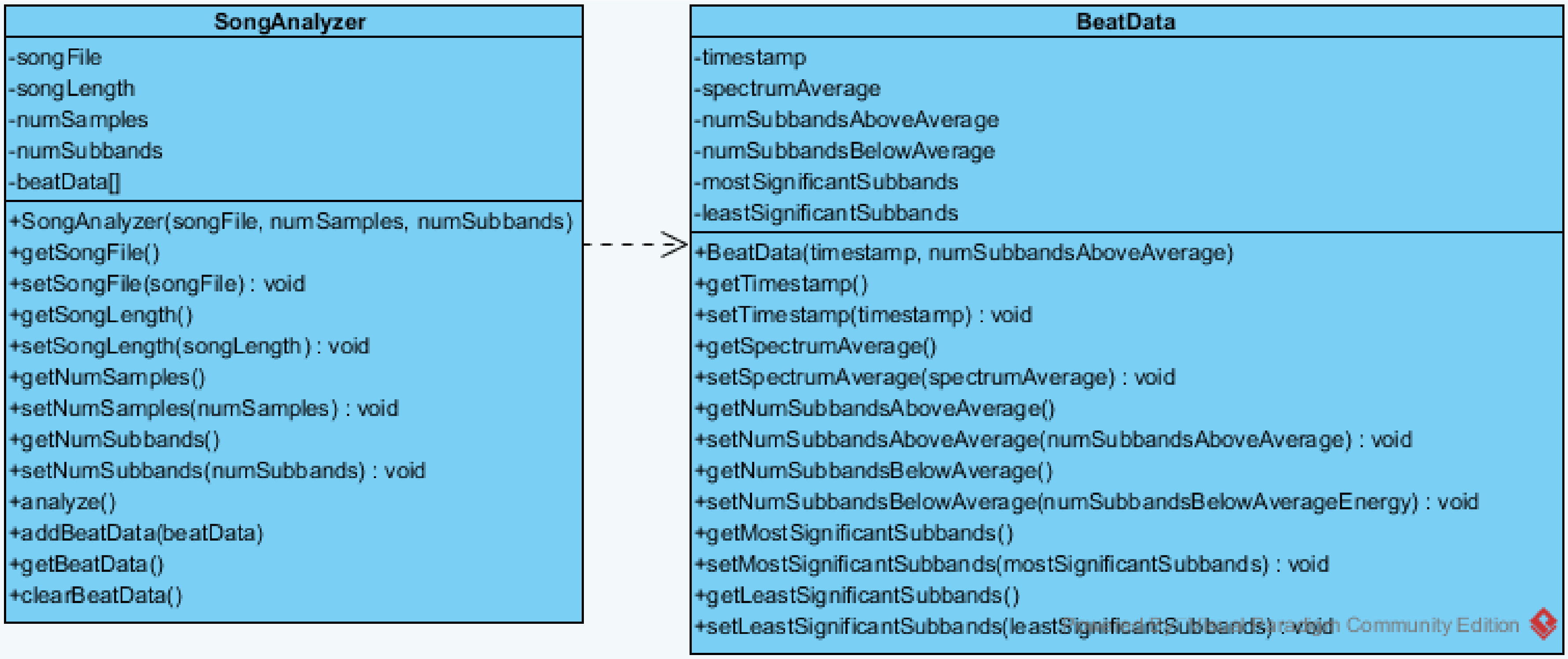


Table of contents

- Overall status
  - Tasks finished
  - Tasks open
- Development
- Beat Detection Algorithm
  - Architecture idea
  - Challenges
  - Actual functionality
- Visual Outcome
- Demo
- Questions

Beat Detection Algorithm - Architecture



SongAnalyzer

-songFile
-songLength
-numSamples
-numSubbands
-beatData[]

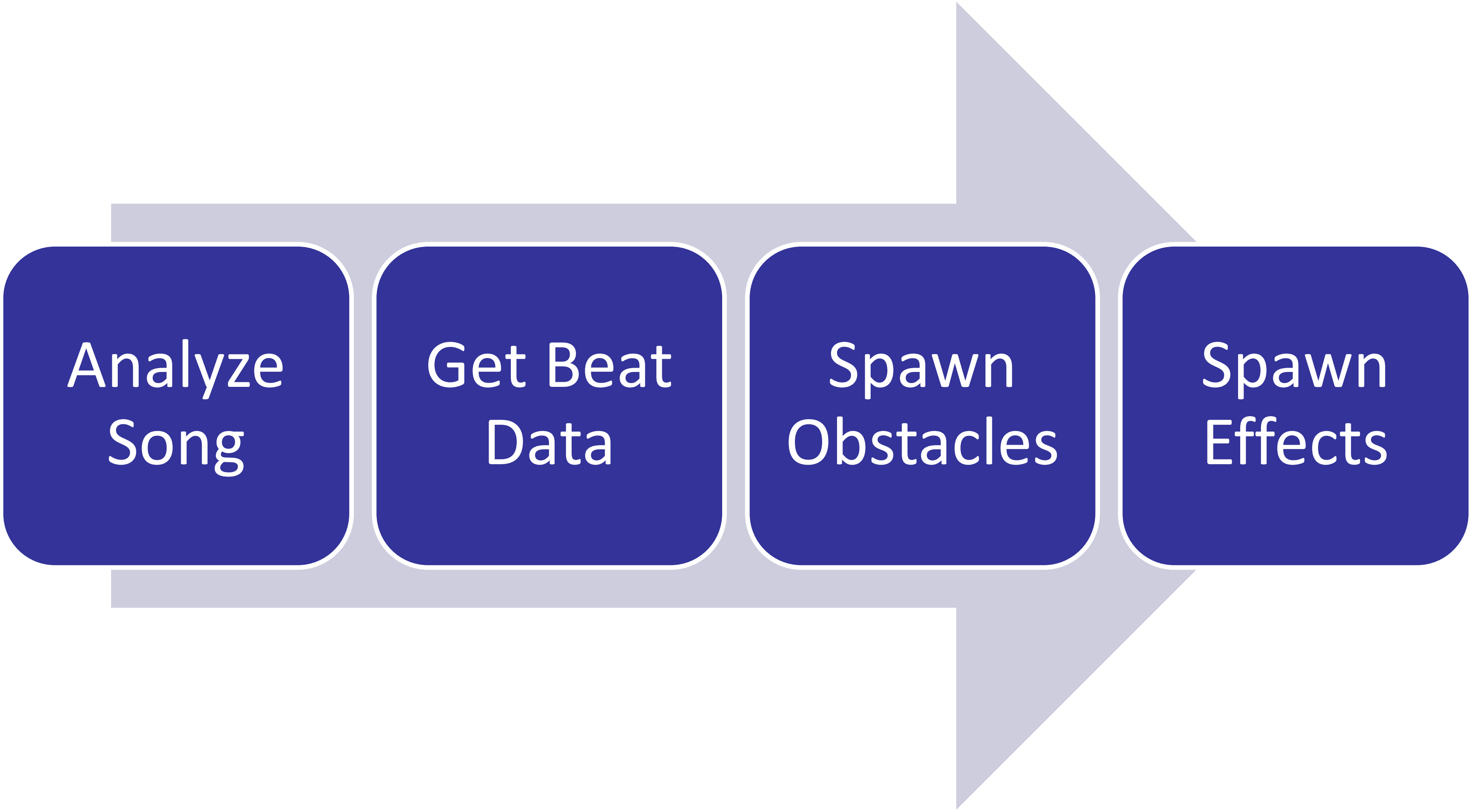
+SongAnalyzer(songFile, numSamples, numSubbands)
+getSongFile()
+setSongFile(songFile) : void
+getSongLength()
+setSongLength(songLength) : void
+getNumSamples()
+setNumSamples(numSamples) : void
+getNumSubbands()
+setNumSubbands(numSubbands) : void
+analyze()
+addBeatData(beatData)
+getBeatData()
+clearBeatData()

BeatData

-timestamp
-spectrumAverage
-numSubbandsAboveAverage
-numSubbandsBelowAverage
-mostSignificantSubbands
-leastSignificantSubbands

+BeatData(timestamp, numSubbandsAboveAverage)
+getTimestamp()
+setTimestamp(timestamp) : void
+getSpectrumAverage()
+setSpectrumAverage(spectrumAverage) : void
+getNumSubbandsAboveAverage()
+setNumSubbandsAboveAverage(numSubbandsAboveAverage) : void
+getNumSubbandsBelowAverage()
+setNumSubbandsBelowAverage(numSubbandsBelowAverageEnergy) : void
+getMostSignificantSubbands()
+setMostSignificantSubbands(mostSignificantSubbands) : void
+getLeastSignificantSubbands()
+setLeastSignificantSubbands(leastSignificantSubbands) : void

Beat Detection Algorithm - Architecture



Analyze Song

Get Beat Data

Spawn Obstacles

Spawn Effects

Beat Detection Algorithm - Challenges

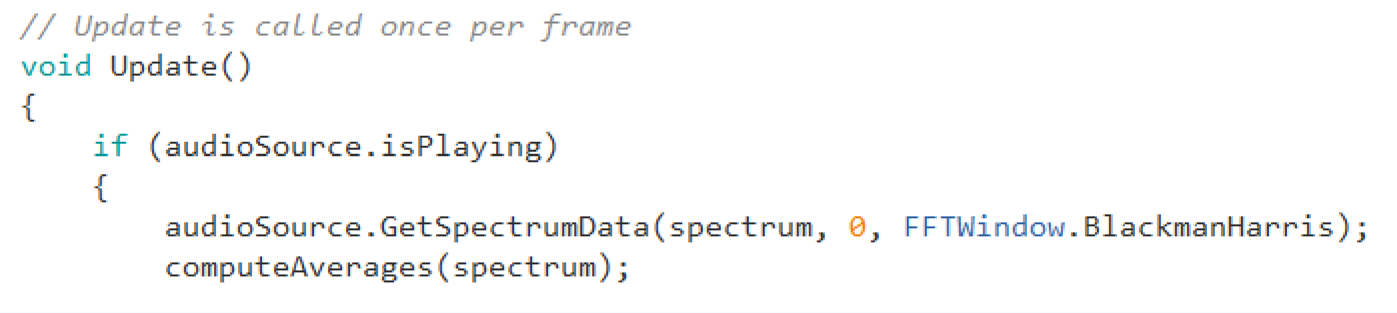
- Different APIs had very different pros and cons.
  - Offline processing vs. on-stream processing.
  - Difficult / incompatible implementations.
  - Weak out-of-the-box functionalities vs. high complexity.

- A lot of trial and error was required.
  - Different sampling methods.
  - Different comparing methods.
  - Different APIs.

- Not all genres work equally good with equal parameters.

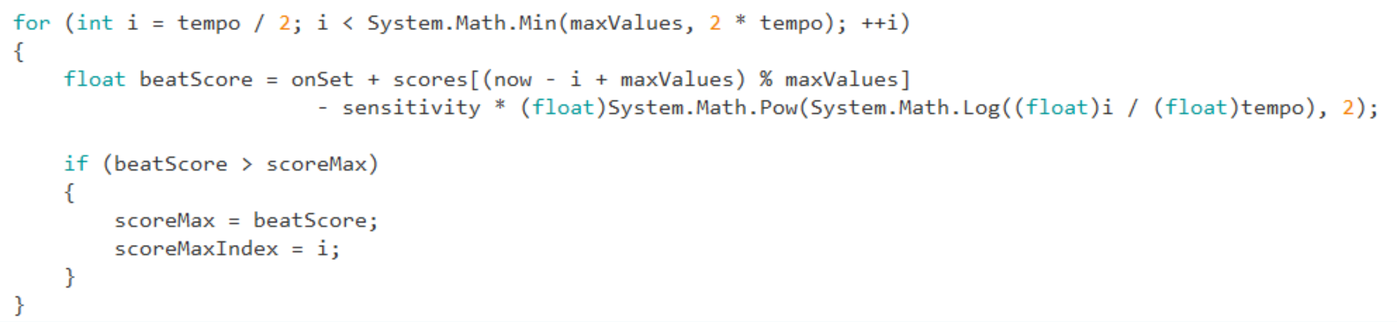
Beat Detection Algorithm – Actual functionality

- Use native Audio Library from Unity:



```
// Update is called once per frame
void Update()
{
    if (audioSource.isPlaying)
    {
        audioSource.GetSpectrumData(spectrum, 0, FFTWindow.BlackmanHarris);
        computeAverages(spectrum);
```

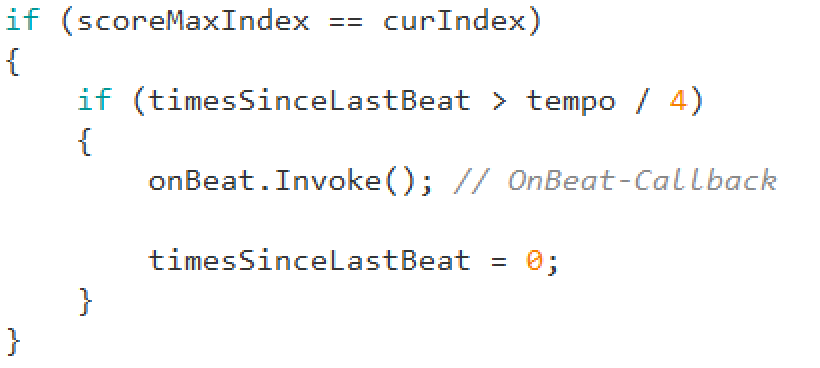
- Search for possible peak value:



```
for (int i = tempo / 2; i < System.Math.Min(maxValues, 2 * tempo); ++i)
{
    float beatScore = onSet + scores[(now - i + maxValues) % maxValues]
                        - sensitivity * (float)System.Math.Pow(System.Math.Log((float)i / (float)tempo), 2);

    if (beatScore > scoreMax)
    {
        scoreMax = beatScore;
        scoreMaxIndex = i;
    }
}
```

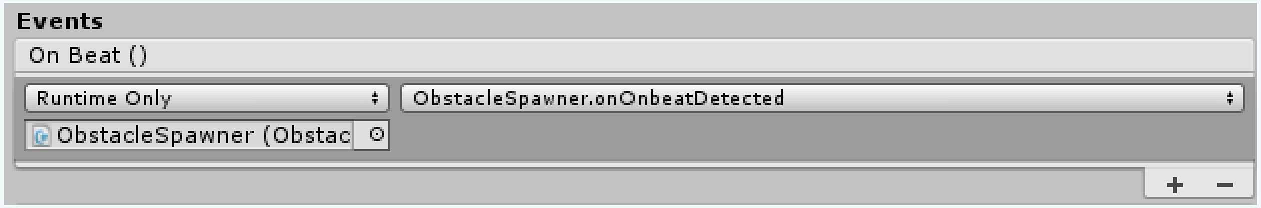
Beat Detection Algorithm – Actual functionality

- Trigger Callback if we have a peak, and if the last one wasn‘t to recent:



```
if (scoreMaxIndex == curIndex)
{
    if (timesSinceLastBeat > tempo / 4)
    {
        onBeat.Invoke(); // OnBeat-Callback

        timesSinceLastBeat = 0;
    }
}
```

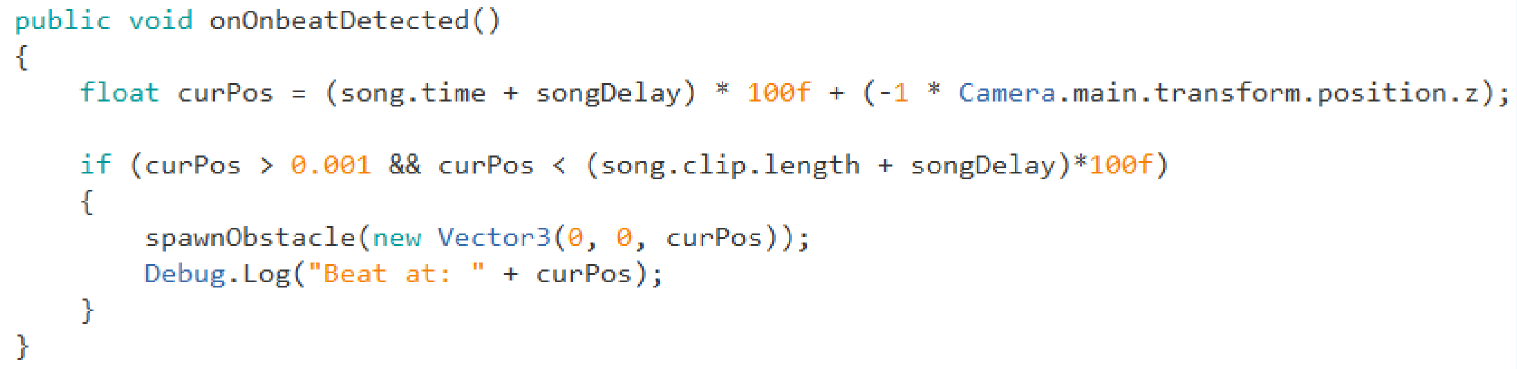
- Add callback function:



Events
On Beat ()
Runtime Only | ObstacleSpawner.onOnbeatDetected
ObstacleSpawner (Obstac

Beat Detection Algorithm – Actual functionality

- Function called in the obstacle spawner:



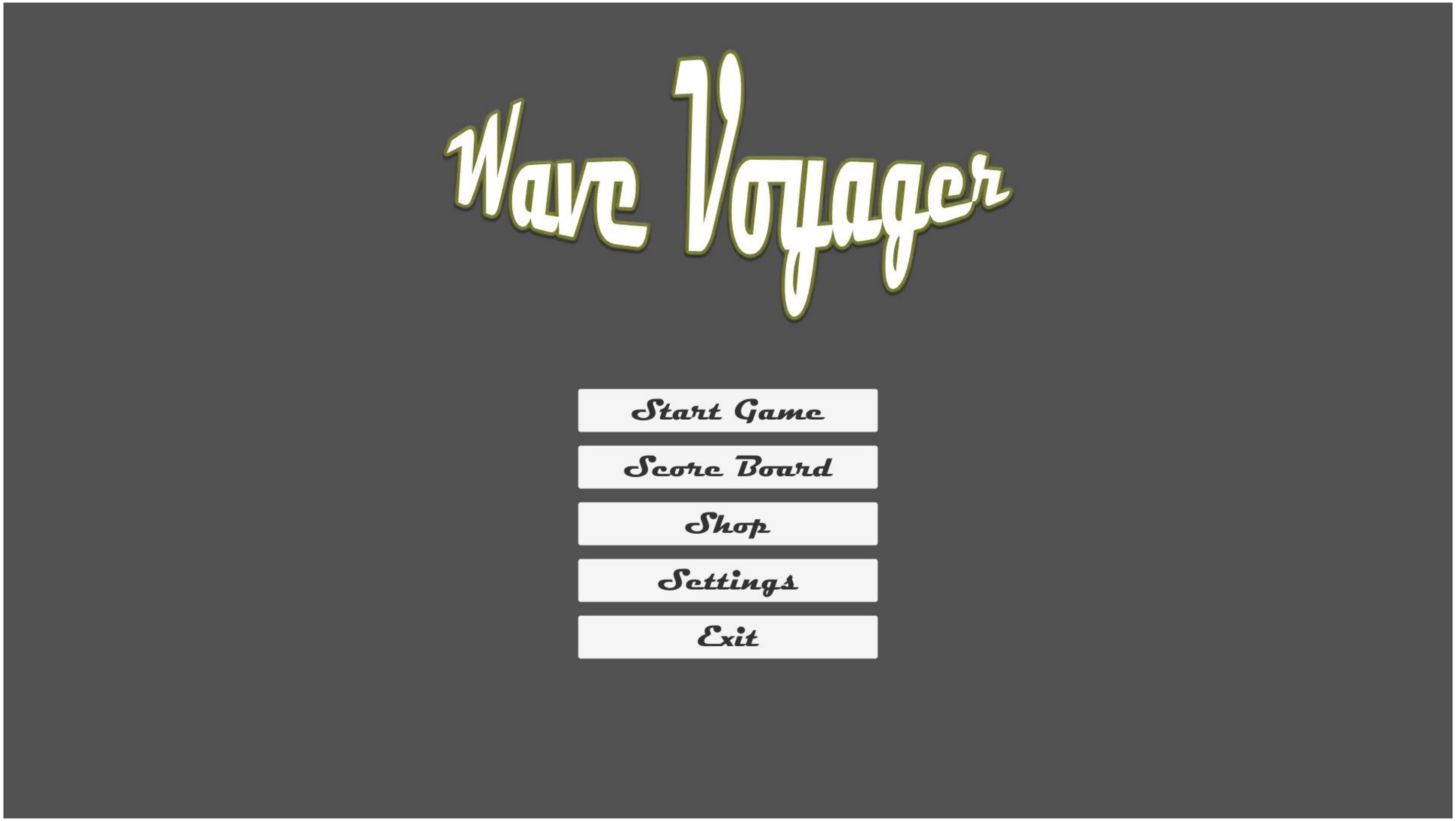
```
public void onOnbeatDetected()
{
    float curPos = (song.time + songDelay) * 100f + (-1 * Camera.main.transform.position.z);

    if (curPos > 0.001 && curPos < (song.clip.length + songDelay)*100f)
    {
        spawnObstacle(new Vector3(0, 0, curPos));
        Debug.Log("Beat at: " + curPos);
    }
}
```

- Song is played silently in the background for processing.
    - Obstacles are being spawned.
- Actual Song starts 5s later.

Table of contents

- Overall status
  - Tasks finished
  - Tasks open
- Development
- Beat Detection Algorithm
  - Architecture idea
  - Challenges
  - Actual functionality
- Visual Outcome
- Demo
- Questions

Visual Outcome



Visual Outcome

Table of contents

- Overall status
  - Tasks finished
  - Tasks open
- Development
- Beat Detection Algorithm
  - Architecture idea
  - Challenges
  - Actual functionality
- Visual Outcome
- Demo
- Questions

Demo

DEMO

Table of contents

- Overall status
  - Tasks finished
  - Tasks open
- Development
- Beat Detection Algorithm
  - Architecture idea
  - Challenges
  - Actual functionality
- Visual Outcome
- Demo
- Questions