Design Brief

Taiko-Inspired Rhythm Game

Andrew Nguyen

18th of March, 2025

Software Development

Units 3 & 4

Mr. Toet

Table of Contents

1. Problem Definition

1.1 Overview

The solution is a rhythm game based off of the existing popular Japanese video game *Taiko no Tatsujin* created and published by Bandai Namco.

1.2 Current Solution Inefficiencies

Taiko no Tatsujin and its variants are played through the physical arcade machine, an official console port, or on a desktop machine.

On a desktop machine, there are a few main variants of the game.

Many of these variants are simulators, which seek to recreate the official arcade version created by Bandai Namco as closely as possible, using the TJA (.tja) file format to store charts (maps) for playable songs emulating Namco's proprietary chart format. Many official charts have been ported to TJA format and and are thus playable on simulators supporting it.

Some *Taiko no Tatsujin* simulators include:

- *TJAPlayer3* (the base for most modern simulators, originally developed by AioiLight)
- *OpenTaiko* (the most recent fork of TJAPlayer, maintained by 申しコミ, or 0AuBsq)
- *TaikoUnity* (now defunct, community complaints about audio lag were common)
- *Taiko Web* (open-source, web-based simulator, developed by Bui)

Many simulators are built on *TJAPlayer*. *TJAPlayer* itself was built on *DTXMania*, a simulator for the Konami game *Drummaster*.

A popular variant of *Taiko no Tatsujin* (although officially unaffiliated with the Namco franchise) is *osu!taiko*, a separate game mode within the larger *osu!* rhythm game. osu!taiko is not a simulator, and uses its own format (.osz) to store charts for playable songs. *osu!taiko* has the largest and most active community of *Taiko* players despite being unaffiliated with the official franchise.

The primary issue with the current solutions is that their file formats for their charts are not cross-compatible. *osu!taiko* is the most popular *Taiko* game, however it is not compatible with the simulator format (.tja), which many official songs created by Namco are created with. Likewise, *Taiko no Tatsujin* desktop simulators such as *TJAPlayer* are not compatible with the osu!taiko format (.osz).

This is with the exception of *Taiko Web*, which is able to play both osu! and TJA formats. *Taiko Web*, being web-based, is still quite unstable and susceptible to input lag, worsening the gameplay experience on weaker/older machines, especially ones with a weak GPU. It also makes the process of importing TJA and osu! files slow and cumbersome, as there is no directory to easily drag TJA and/or osu! files to.

2. Proposed Users

2.1 Primary User

Stephanie Li

Relationship: Friend

Occupation: Student

Age: 16

Location: Melbourne, Australia

Client Profile Description:

Stephanie is the developer's friend, who is currently a student at Balwyn High School. She plays *osu!taiko f*requently (primarily on keyboard), and is currently ranked #13,524 globally and #245 in Australia. She also plays the Taiko simulator *OpenTaiko* to practise official maps. She owns two laptops, one for personal use and one for school, with high and medium specifications, respectively. She also owns an officially licensed HORI drum controller. She has a moderate level of familiarity with computer software.

Solution Permissions:

Stephanie will not have any elevated permissions within the final solution, although certain tools such as a developer console may be enabled in indev builds sent to her allowing for easier playtesting.

2.2 Secondary User(s)

Kaitlyn Young

Relationship: Friend

Occupation: Student

Age: 16

Location: Melbourne, Australia

Client Profile Description:

Kaitlyn is the developer's friend, who is currently a student at Balwyn High School. She plays a variety of rhythm games and first introduced the developer to the *Taiko no Tatsujin* franchise. She owns a school-provided laptop with limited capabilities. Kaitlyn does not have a great deal of familiarity with computer software, but can use her laptop with relative ease.

Solution Permissions:

Kaitlyn will not have any elevated permissions within the final solution, although certain tools such as a developer console may be enabled in indev builds sent to her allowing for easier playtesting.

Andrew Nguyen

Occupation: Student

Age: 16

Location: Melbourne, Australia

Client Profile Description:

Andrew is the developer of the solution, and is currently a student studying Software Development 3/4 at Balwyn High School. Andrew plays osu!taiko and *Taiko no Tatsujin* frequently both at home (on keyboard as well as on a drum controller) and on physical arcade machines, and he is currently ranked #6,912 globally and #125 in Australia for *osu!taiko.*

Solution Permissions:

Andrew, as the developer, will have access to the developer console to allow for easier debugging and playtesting.

3. Program Details

3.1 Programming Language

The solution will be built on the Unity game engine, which uses C# for its built-in Unity Scripting API.

3.2 Justification

The Unity game engine was chosen for the solution's development for a number of reasons, including stability, cross-platform compatibility, and developer expertise.

The existing *Taiko* solutions (with the exception of osu!taiko) are often unstable and several years old, being built on old open-source frameworks. *Taiko Unity* (a now-defunct *Taiko* simulator built on the Unity game engine), did not suffer the same stability issues that the older frameworks had, but did receive complaints about audio lag, which is very important to a user's rhythm game experience.

The input system in the Unity engine also allows for controller/gamepad compatibility, allowing for the use of certain devices such as a DualSense PS5 controller or a drum such as that of the official Nintendo-licensed HORI drum controller.

The Unity game engine also presents future opportunities for deployment to a variety of different platforms, as it allows for builds on Windows, Mac OS X, Linux, iOS, Android and WebGL. This allows for extremely efficient deployment to a number of platforms with minimal effort.

The developer has several years of experience with the Unity game engine and C#, having worked on many personal projects since 2018, including several rhythm games. Because of this prior experience, the developer knows how to minimise or resolve certain issues pertaining to audio lag.

Extensive documentation and support is available for the Unity game engine, having existed since 2005. This is the primary reason why an alternative game engine such as Godot was not considered, as Godot is much younger and does not yet offer the same level of support and documentation comparable to that of the Unity engine.

3.3 Data Structures

3.3.1 Input

Unity takes in user input data directly from peripherals using a built-in Input class. (Unity has recently debuted a new and improved Input system, so this may no longer apply. A plugin such as Rewired may also be used by the developer to allow for input remapping and ease of controller compatibility.) Input taken in from the user will include:

- Keyboard inputs for menu navigation and gameplay
- Mouse inputs (click events) for menu navigation
- Gamepad inputs for menu navigation and gameplay

For storing and reading charts (maps/songs), the solution will allow both osu! beatmap files (.osz) and TJA (.tja) files as inputs.

A TJA file is a plaintext file which contains a variety of metadata (including song title, a linked audio file, beats per minute, and difficulty) and hexadecimal data.

An .osz file is a file containing zip-compressed data containing beatmaps and additional resources such as skin data, music, and additional map variations based on difficulty.

The solution will also read audio files including .mp3, .ogg, and .wav files.

3.3.2 Internal Storage

Map (playable song) data will be kept in a Map class, partially based on the existing .osu format.

Map data:

- Title (string) - Romanised song title
- TitleUnicode (string) - Song title in Unicode
- Artist (string) - Artist name
- ArtistUnicode (string) - Artist name in Unicode
- Creator (string) - Map creator
- Version (string) - Name of difficulty
- Source (string) - Song’s original source media
- HitObjects (List<HitObject>) - List of HitObjects to be read during the map’s runtime.

- SourceClip (AudioClip) - The song's audio file.
- Background (Texture2D) - Background to use during gameplay.

Gameplay:

- Score (integer) - The player's current score.
- JudgeGood (int) - The number of "Good" hits (inputs within 0-50 ms of the note)
- JudgeOkay (int) - The amount of "Okay" hits (inputs between 51-120 ms of the note)
- JudgeMiss(int) - The number of misses (notes failed to hit within 120 ms)
- Accuracy (float) - The calculated average "accuracy" of the timing of the player's inputs.
- Combo (integer) - The number of successful consecutive inputs.
- CurrentTime (float) - The current playback position within the map.
- Progress (float) - How much of the map has been completed (in a percentage).
- Retries (integer) - The number of times the player has retried the map.
- IsPaused (boolean) - If the game is paused or not.

3.3.3 Output

The gameplay will involve the solution reading from a linked TJA or osu! beatmap file to be outputted to the user interface, for both the solution's map selection screen as well as gameplay.

During gameplay, users will see real-time and responsive visual and audio feedback such as hit judgments (e.g. "Good," "Okay," "Miss") based on their keyboard inputs in relation to the note timing of the map. The gameplay interface will also include live score and combo counter (displaying the number of successful consecutive hits), a progress bar, and an accuracy percentage display, and some background visuals (read from the inputted Map data).

Live audio feedback will also be played according to user inputs, potentially adding custom hitsounds.

Performance metrics and user highscores may be saved to a JSON file or a similar file format.

4. Feasibility

4.1 Economic Feasibility

There are very little economic constraints that will hinder the development of the software solution. All tools and software used to assist the development of the solution are free to use. There do exist paid versions for the Unity game engine, however the free Personal License is being used during development. Plug-ins and assets used to assist and accelerate development (such as the industry standard Rewired for input) may have additional costs attached to them, however are often one-time payments and do not require a subscription. No additional economic constraints are brought on by the use of paid plugins aside from the cost upfront.

The developer may purchase the Rewired plugin to assist development, meaning there will be a small economic cost during development.

4.2 Technical Feasibility

The solution is being primarily developed on two of the developer's devices, using GitHub as a means of syncing the project between them. The primary device is a desktop machine which the developer has used for all prior development projects. The secondary device is a laptop which the developer uses primarily for school and work purposes.

Primary device specifications

- CPU: Intel Core i9-10900 CPU @ 2.80GHz
- Operating system: Windows 10
- Memory: 32GB RAM
- GPU: NVIDIA GeForce RTX 3060
- Storage: 4TB

Secondary device specifications

- CPU: AMD Ryzen 5
- Operating system: Windows 11
- GPU: Integrated AMD Radeon Graphics
- Memory: 16GB
- Storage: 1TB

Both machines are able to smoothly support development using the tools provided. As the solution is not render-intensive, a strong GPU is not needed. The bulk of the development will be handled on the primary desktop machine due to its much stronger specifications. Development and playtesting will intermittently be done on the secondary device to ensure weaker machines are able to run the software solution.

Both machines also run Windows. This makes developing and playtesting for alternative platforms such as MacOS and Linux will be difficult. Games developed in Unity are able to be built for different operating systems (including MacOS, Linux, iOS, Android and more) with additional packages. Due to the proposed users only running Windows devices, it is unlikely that deployment to alternative platforms will be necessary. If, at a future point in development, deployment to a different platform is necessary, extensive playtesting will be required on those platforms to ensure the solution continues to function as intended.

The developer is quite experienced with game development using Unity and C# specifically. This makes the development of the solution substantially more feasible and efficient, as no new tools have to be learned within the development time granted to the developer. Additional open-source code that may be included in the solution will also have to be compatible with C# and Unity.

4.3 Scheduling Feasibility

To ensure the project reaches completion, it is important for a proper and appropriate schedule to be maintained throughout development.

The developer has a variety of other time commitments that may hinder the speed at which the solution is developed. The developer, as a student, is currently undertaking six VCE studies, two of which are Unit 3/4 subjects (Software Development and Media). Media, as a separate folio subject with a very large time commitment, alongside the content-heavy Mathematical Methods and Specialist Mathematics, will take time away from development. Therefore, the project will have to be scheduled carefully to ensure the solution can be feasibly developed within the time allocated.

Consultations with the proposed users will also need to be carried out regularly, who all have their own individual time commitments as well. This will require rigid scheduling on the part of the developer to arrange these meetings effectively.

5. Originality

5.1 Focus

The proposed solution stands out as it combines and unifies elements and capabilities from existing solutions, as well as resolves several issues with the other solutions including file format compatibility, input lag, and general gameplay stability.

Existing solutions are mostly constrained by their file format support. The proposed solution seeks to natively handle both osu! beatmap files (.osz) and TJA (.tja) files. The process of importing songs will also be streamlined with minimal file management skills being required to do so (unlike the popular simulator *TJAPlayer* and its variants).

The solution, being built on modern game development frameworks (including the Unity game engine), will also be highly optimised for a number of different platforms. The now-defunct *Taiko Unity,* which was also made in the Unity game engine, suffered greatly from gameplay stability issues and audio lag. The proposed solution seeks to resolve all issues with stability and performance that previous iterations of *Taiko Unity* suffered from.

5.2 Comparison

Existing *Taiko no Tatsujin* simulators are built on old frameworks (especially *TJAPlayer*, itself built on a different simulator for a different game altogether), which ultimately struggle with performance and cross-platform compatibility. The proposed software solution, using the modern Unity game engine, will offer a much more stable user experience as well as be playable on different platforms

The *osu!* platform, while having cultivated a very large community of *Taiko* players through the gamemode *osu!taiko*, its proprietary file format that the platform uses to store content ultimately limits access to the rest of the *Taiko no Tatsujin* content landscape.

Taiko Web does attempt to support multiple file formats, however its platform limits it quite heavily. It is not playable on mobile, gamepad compatibility is limited, and there is a considerable instability on older devices with regards to responsiveness, input lag and audio lag.

This solution therefore offers a stable platform that combines the extensive content libraries of the main simulators as well as *osu!taiko,* overcoming many technical limitations of the other solutions, and a much more stable, smooth and intuitive user experience.