

Detailed Game Specification: 8-Bit Hero

Course: COMP 2659, Winter 2023

Instructor: Paul Pospisil

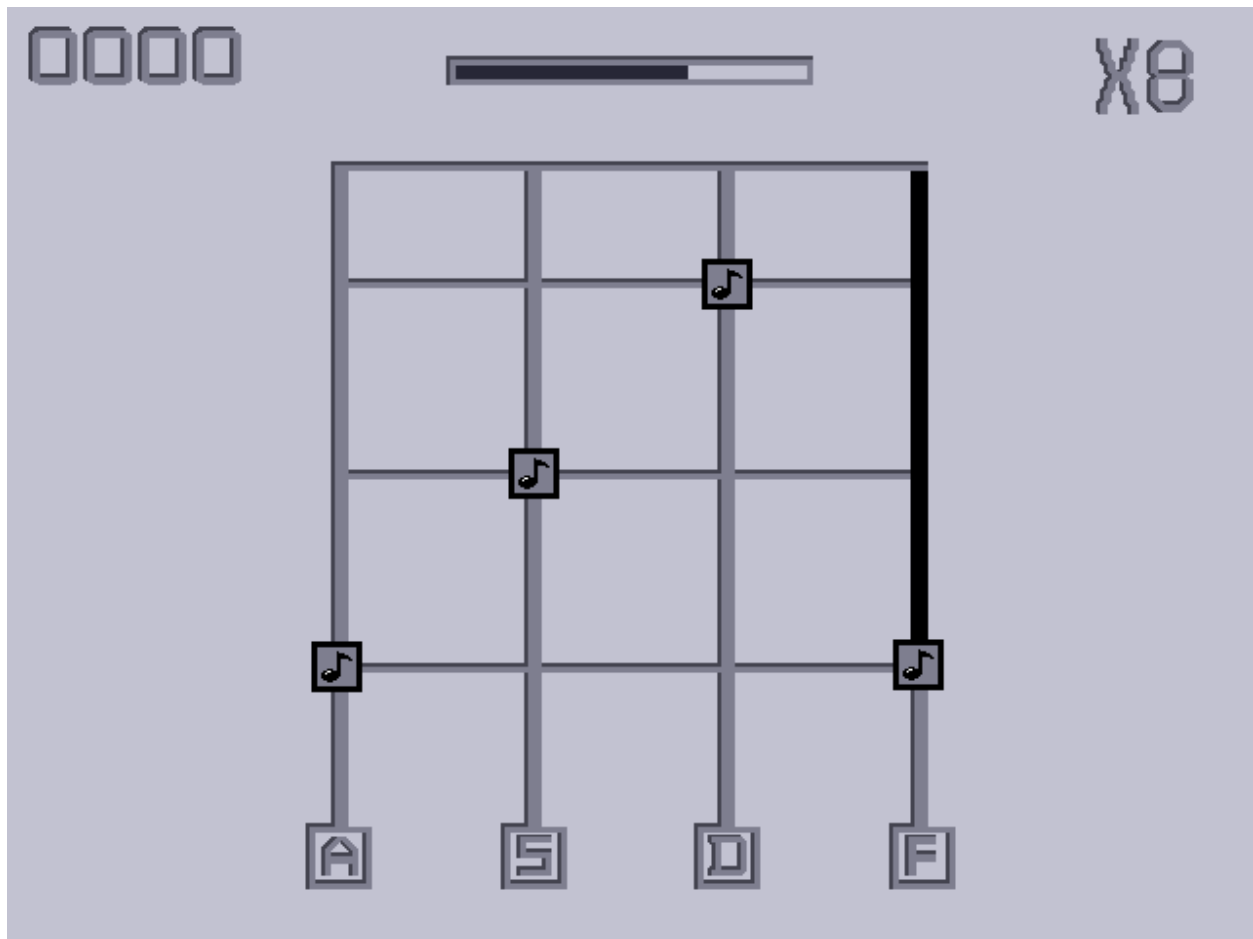
Authors: Andrew Boisvert & Kyle Scidmore

Last Modified: Jan. 15, 2023

1. General Game Overview

8-Bit Hero is a competitive, one or two-player, music-based rhythm game that lets players simulate playing a guitar to chiptune music. The game is viewed from a top-down perspective and features a “virtual fretboard” as the main playing interface. At the top of the screen the player’s score, score multiplier and fail bar are posted. The virtual fretboard is a series of vertical lines with small rectangles at the bottom which represent the frets that you can play. Thin horizontal lines cross the vertical lines and represent downbeats (1s and 3s). As chiptune music is playing, small rectangular blocks which are called “notes” begin moving down the vertical lines of the fretboard from the top of the screen. The game is timed so that the note will line up with its respective fret at the correct timing and the player must then press the button for that fret to play the note and get the point(s) for it. Points are collected for each correct note that is played. Each consecutive note that you play begins the building of a note streak. Note streaks increase the multiplier for every 10 notes that are played in a row.

In one-player mode, the player attempts to score as high as possible on a song by hitting as many notes as they can in succession. In two-player mode, players take turns playing the same song and whoever achieves the highest score is the winner. Points are scored for every note that is played at the correct time. Points vary depending on the type of note(s) that are played: 1 point for notes, 2 points for chords, and 3 points for long notes/chords.



2. Game Play Details for Core 1-Player Version

Objectives and Rules

A game round starts with the player looking at an empty fretboard. At the start of the round the player starts with a score of 0, a score multiplier of 1x, and the music begins playing. When the music starts playing, notes begin to move downwards from the top of the screen at a velocity that is synchronized with the music so that notes reach the frets on their appropriate counts. The objective of the game is to achieve the highest score possible at the end of the round.

A point is scored when a player correctly plays a note when it is within the note's scoring boundary. The note's scoring boundary is the outer boundary of the fret and the game checks if the note's center pixel is within the scoring boundary. Different scores are available for different note types. There are different note types: basic notes, chords, and long notes. Basic notes are single notes played individually and are worth 1 point. Chords are 2-3 notes played together and are worth 2 points. Long notes are either single notes or chords that are pressed and held before release and are worth 3 points.

A score multiplier adjusts the points that you score. Note streaks are recorded, keeping track of how many notes the player correctly plays in a row, and the score multiplier increases for every 10 notes

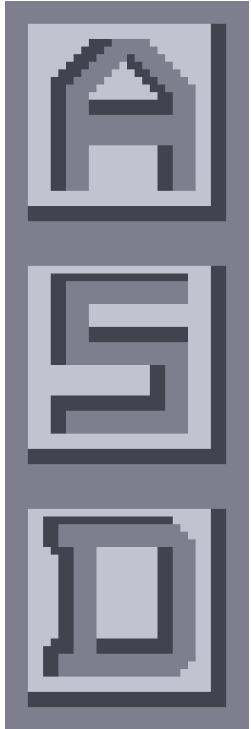
added to the note streak. Streaks of 10 after achieving a streak of 30 will not affect the multiplier. The score multipliers are 1x, 2x for note streaks between 10 and 20, 3x for note streaks between 20 and 30, and 4x for note streaks greater than 30. Failed notes reset the multiplier because they are based upon the note streak.




A fail bar, which starts 50% full, is located at the top middle of the display. If the player correctly plays a note, the fail bar will tick up by 8 pixels. Likewise, if a player misses a note or plays a fret with no note on it, the fail bar will tick down by 8 pixels. If the fail bar is completely depleted, the player fails the song and must restart.



Note Physics, Note and Fret Collisions

The velocity of the notes in the game is calculated by the tempo of the song that is selected so that the notes line up with the frets on-beat. This is calculated based upon the clock speed of the MC68000. Collisions between frets and notes are monitored to determine if pressed frets are within the scoring boundary or if notes have been missed.

Objects

<u>Object or Object Type</u> <u>Name</u>	<u>Properties</u>	<u>Behaviours</u>	<u>Graphical Image</u>
Fret	<ul style="list-style-type: none"> position (const integer pair: x, y) size (integer pair: 32, 32) 	<ul style="list-style-type: none"> event 	

			
Note	<ul style="list-style-type: none"> • position (integer pair: const x, y) • vertical velocity (integer, based on song tempo) • vertical direction (negative integer) • size (constant integer pair: 32, 32) 	<ul style="list-style-type: none"> • move • event • generate 	
Score	<ul style="list-style-type: none"> • position (constant integer pair: x, y) • entire size (constant integer pair: 128, 32) • digit size (constant integer pair: 32, 32) 	<ul style="list-style-type: none"> • update 	

	<ul style="list-style-type: none"> • value (positive integer) 		
Score Multiplier	<ul style="list-style-type: none"> • position (constant integer pair: x, y) • entire size (constant integer pair: 64, 32) • digit size (constant integer pair: 32, 32) • value (positive integer) 	<ul style="list-style-type: none"> • update 	
Fretboard	<ul style="list-style-type: none"> • position (constant integer pair: x, y) • size (constant integer pair: height, width) 		<p>featured in the main grid in the full screenshot</p>
Fail Bar	<ul style="list-style-type: none"> • position (constant integer pair: x, y) • size (constant integer pair: 136, 16) • value (positive integer) 	<ul style="list-style-type: none"> • update 	

Asynchronous (Input) Events

<u>Event Name</u>	<u>Triggering Input Event</u>	<u>Description</u>
Play 1st fret	'A' key is depressed	Sets a note to be played on the 1st fret
Play 2nd fret	'S' key is depressed	Sets a note to be played on the 2nd fret
Play 3rd fret	'D' key is depressed	Sets a note to be played on the 3rd fret
Play 4th fret	'F' key is depressed	Sets a note to be played on the 4th fret
Quit	'Esc' key is depressed	Will set the terminate game flag

Synchronous (Timed) Events

<u>Event Name</u>	<u>Trigger Timing</u>	<u>Description</u>
Notes Moving	Based off song tempo	Notes moving down the fretboard correspond to the tempo of the song and move at a rate to sync up with the music

Condition-Based (Cascaded) Events

<u>Event Name</u>	<u>Triggering Condition</u>	<u>Description</u>
Point Increase	Hitting a note	Accurately hitting a note increases score depending on the note type and multiplier
Fail Bar Uptick	Hitting a note	Accurately hitting a note increases the fail bar
Fail Bar Downtick	Missing a note	Missing a note decreases the fail bar
Multiplier Increase	Hitting notes in succession	Hitting 10 notes in succession increases score multiplier (x1/x2/x4/x8)
Multiplier Decrease	Missing a note	Missing a note drops score multiplier back to x1
Song Failure	Fail bar depleted	Depleting the fail bar causes the player to fail and have to restart/end
Match	Note played is true	Check if the played note is on the fret (for chords and long notes will need to time this)

Hypothetical Gaming Session

The game begins with the fretboard displayed with no notes present. The starting score of 0 is displayed just offset from the top-left corner of the display. The starting score multiplier of “x1” is displayed just offset from the top-right corner of the display, and the fail bar is full and displayed in the top middle of the display .

Chiptune music begins playing and some basic notes that synchronize with the music that the player is hearing begin moving down the fretboard. The player misses the first couple of notes and the main melody of the song does not play, instead playing a “tik” sound for each missed note. The player then plays a correct note. When they do so, the main melody begins to play again, 1 point is added to their score on the screen, and their note streak is calculated in the back end of the game. The player then plays an incorrect note, the main melody drops out, and a “bzz” sound plays to tell them they played a wrong note. As the player plays these correct and incorrect notes, the fail bar at the top middle of the screen is going upwards and downwards, respectively. The player starts to get a feel for the game and begins correctly hitting all of the notes coming down the fretboard. Once the player plays the 10th correct note in a row, the score multiplier on the screen updates and is now x2. Every note scored by the player is now worth 2 points.

The chorus of the song arrives, and the player sees a few 2-note chords coming down the fretboard. They miss the first one and when they do so, their score multiplier drops back to x1. They correctly play the other chords and are awarded 2 points for each, and the game calculates their note streak.

They continue playing through the song. They are a slightly above average player and manage to bring their multiplier up to x4, then x8, then x4 again, making mistakes in between and dropping it back down. The song begins to reach its conclusion and the player correctly plays the final notes of the song.

The player has now finished the song. The game ends and a message is printed in the middle of the screen congratulating the player on completing the song along with their score.

The player feels confident having successfully completed the last stage and decides to play again to try to beat their score. Disappointingly, the player’s coordination is all off and they have a lot of trouble playing the notes correctly. The fail bar begins to move further and further downwards, until it reaches empty. Once the fail bar empties, the game ends and a message is printed in the middle of the screen that wishes the player better luck next time.

3. Game Play Details for Core 2-Player Version

In the two-player version of the game, both of the players are human and the game is played on separate computers that are connected to one another. All the rules of the game remain the same. The only differences are that the second player is played by a human instead of the computer, and that the song plays twice with control being passed to the second player for the second round. The objective of the 2-player game is to be the player with the highest score at the end of the round.

4. Sound Effects

<u>Sound Effect Name</u>	<u>Brief Description</u>	<u>Event which Triggers Playback</u>
Selected Song	main song, plays through gameplay	start of game round
Wrong note played	bzz	fret depressed and no collision with note
Note not played at all	tik	no fret depressed when note and fret collide
Gave Over Win	little arpeggio melody representing a little guitar solo or similar	win or reset with game over being true
Game Over Lose	Dun Dun Dunnn	lose the game
Menu Selection	buh-ding (C4->C5 note jump)	select the game play mode from the main menu

The selected song's music determines the gameplay. The game will be developed with a simple song of repeating scales and will possibly be expanded (time permitting) to classic video game music like the Tetris theme. Channel 1 will play the main melody with a square wave, channel 2 the supporting melody with a square wave, and channel 3 the percussion section with the noise generator. Channel 2 and 3 will play constantly, but channel 1 will drop out when the player plays the wrong note or does not play a note at all, in which case it will play a sound effect for the respective type of miss. Channel 1's music will begin to play again once the player correctly plays a note.

5. Additional Features (Time Permitting)

- Expand song from simple melody into a classic game tune
- Have more than one song be available to be played
- Have special sprites and effects for different songs
- Have multiple difficulty modes
- Save high scores, even if it's just in volatile memory for play sessions
- Post the note streak to the screen instead of it being a backend calculation

- Give the fretboard horizontal bars that are based upon the song tempo to give the player a visual indicator of the downbeats of the song/song tempo.