Ento Zero

A Visually Accessible 3D Game

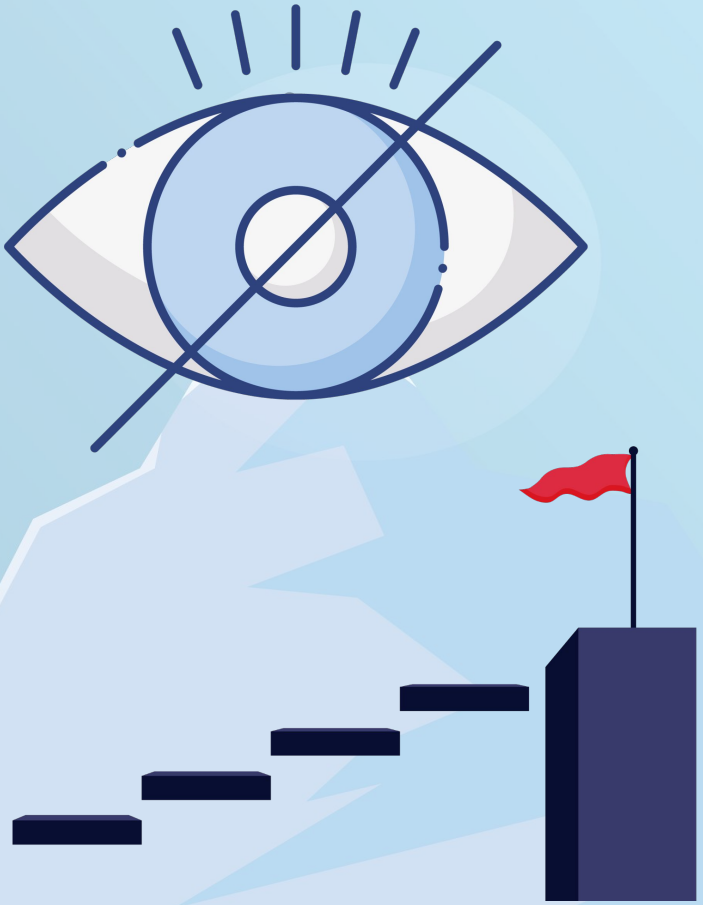
By: Tan Phan, Edward Hear, Matthew Ruest,
Nicholas Heerdt, Chiayene Magbanua, Jose Sison, Zhuohong Lei

Problem

What's the main goal?

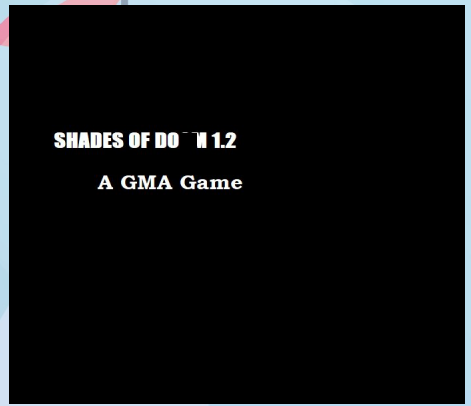
- There are a lack of 3D games that have accessibility for the blind and visually impaired

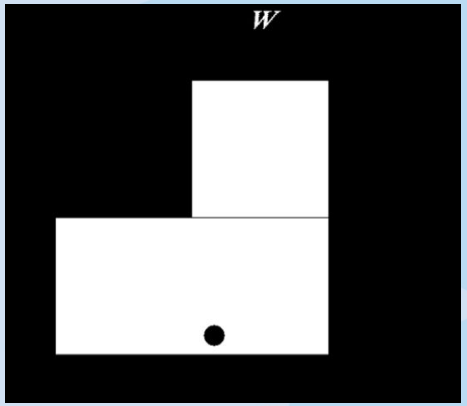
- Making a 3D game visually accessible, fun, and challenging



Competitive Analysis

Shades of Doom





Features

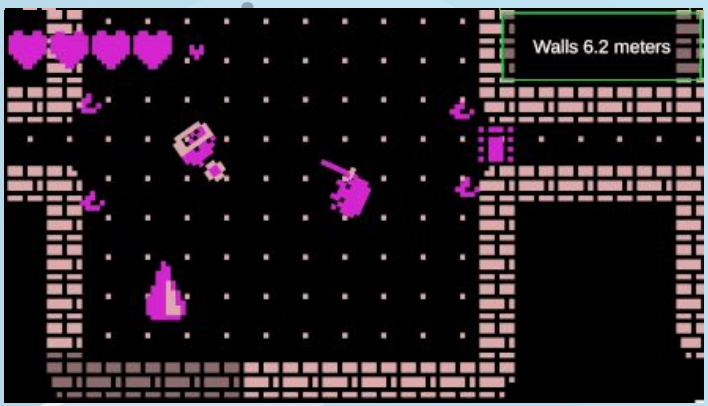
- In-depth positional audio
- Environmental analysis tool
- Numerous assistive controls for blind players

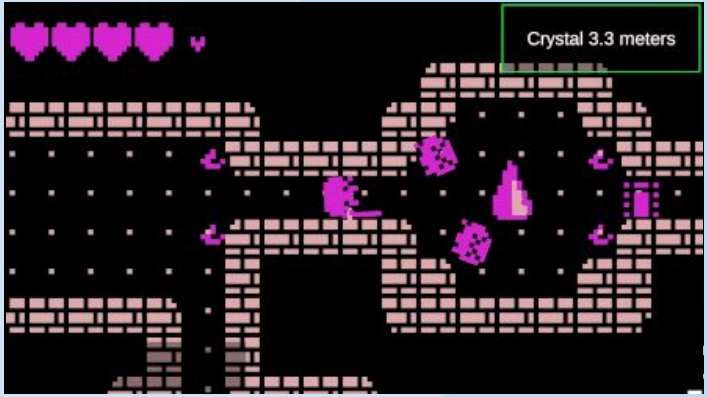
Areas of Improvement

Non-linear levels

Competitive Analysis

Code Dungeon





Features

- Immersive 3D Audio
- Line-of-Sight Tool
- Configuration for controls & audio

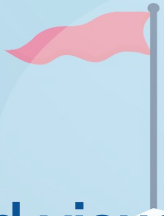
Areas of Improvement

Lack of distinct colors between items and entities

Solution

- Designing features around vision impairment
  - 3D audio with distinct sounds for binaural (both ears) hearing
  - High contrast colors
  - Screen reader compatibility
- Using Quorum game engine for accessibility features

Accessibility



How is our game inclusive to the blind and visually impaired?

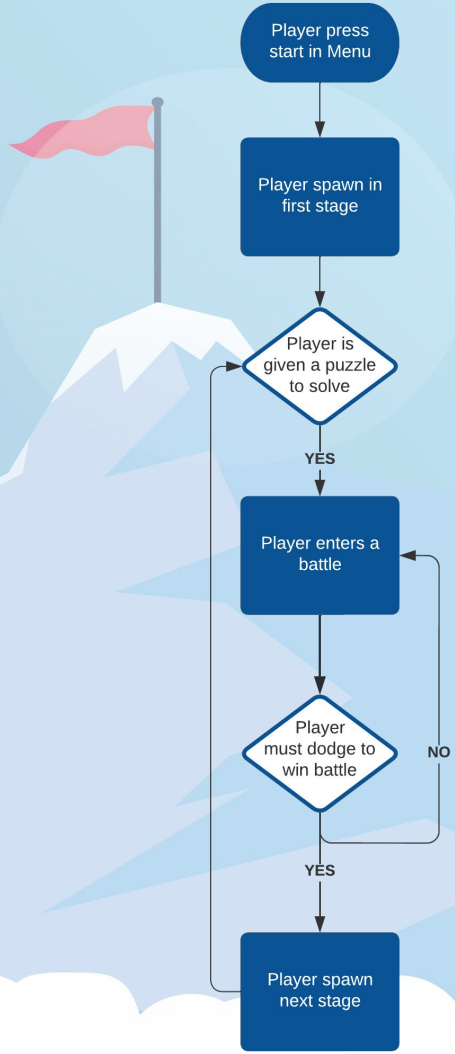
- 3D audio implementation will allow navigation of the levels by users with visual impairments or blindness
- Text and menus will also provide support for screen readers such as NVDA (NonVisual Desktop Access)

Tutorial

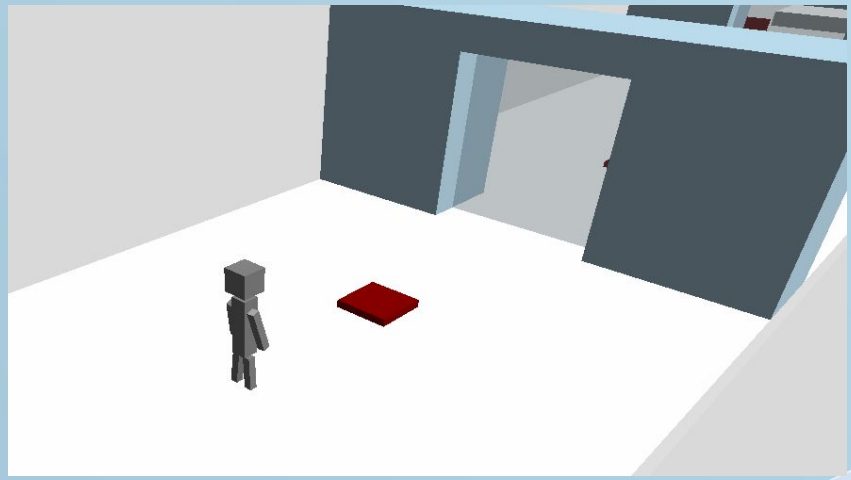
Designed to teach players what each audio cue signifies as well as what the controls are

Design Overview

- Player spawns in the stage
- Given puzzles to solve via auditory cues
- A boss fight occurs at the end leading to the next stage



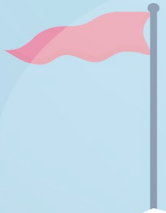
Ento Zero Gameplay



Overworld

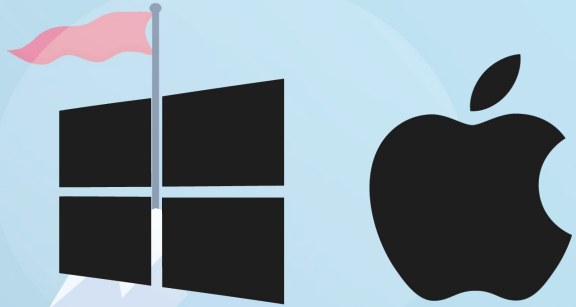
Combat

Functional Requirements



- Menu System - Menu appears at launch of game (provides screen reader compatibility and screen resolution options)
- 3D Audio - Positional audio informing players about surrounding environment
- Interactive Map Assets - Asset for puzzle solving and unlocking new levels
- Autosave System - Automatic way of saving player progression
- Enemy AI - AI for enemy encounters, leading to the combat system
- Combat System - Dodging mechanic for fighting in-game enemies

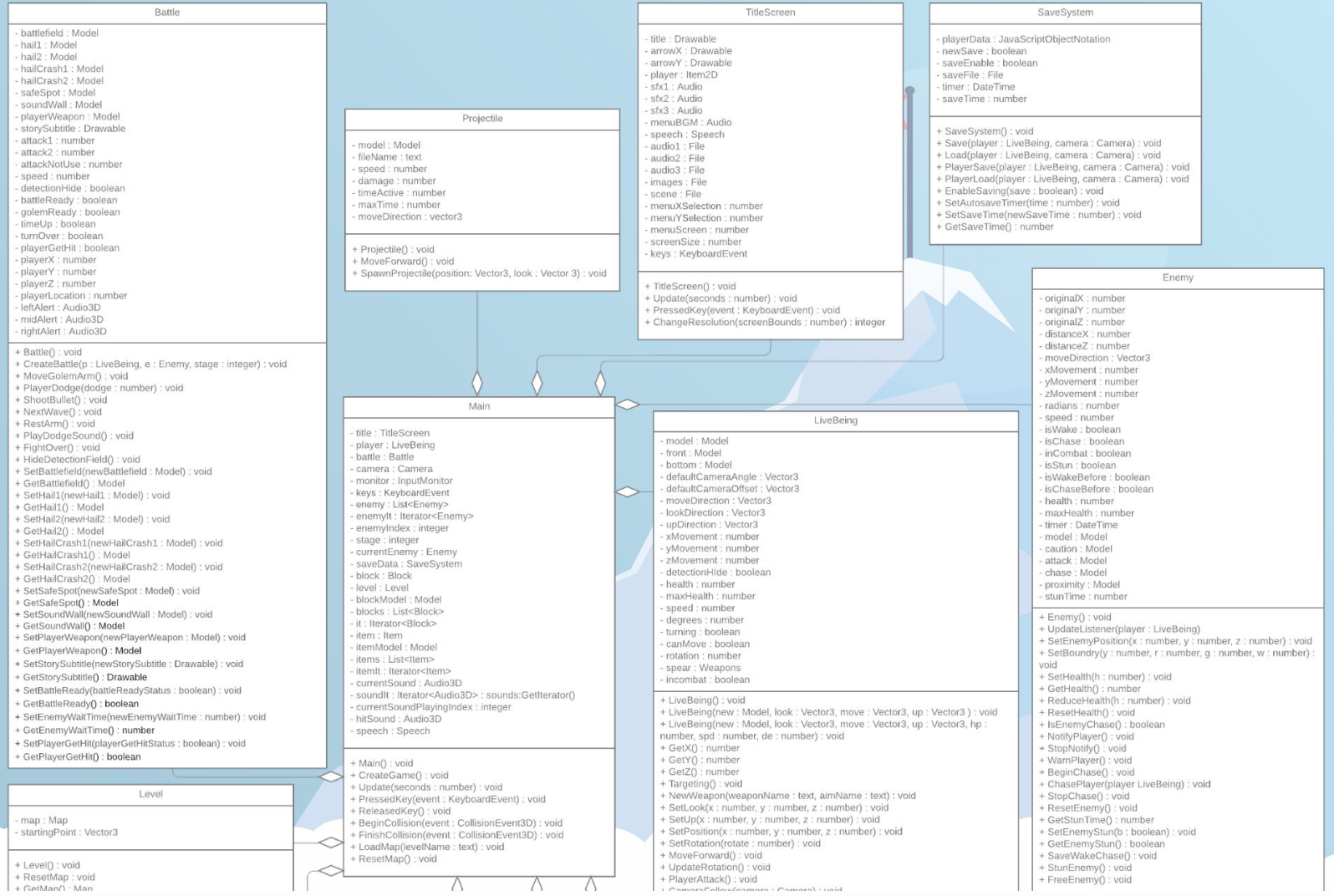
Non-Functional Requirements



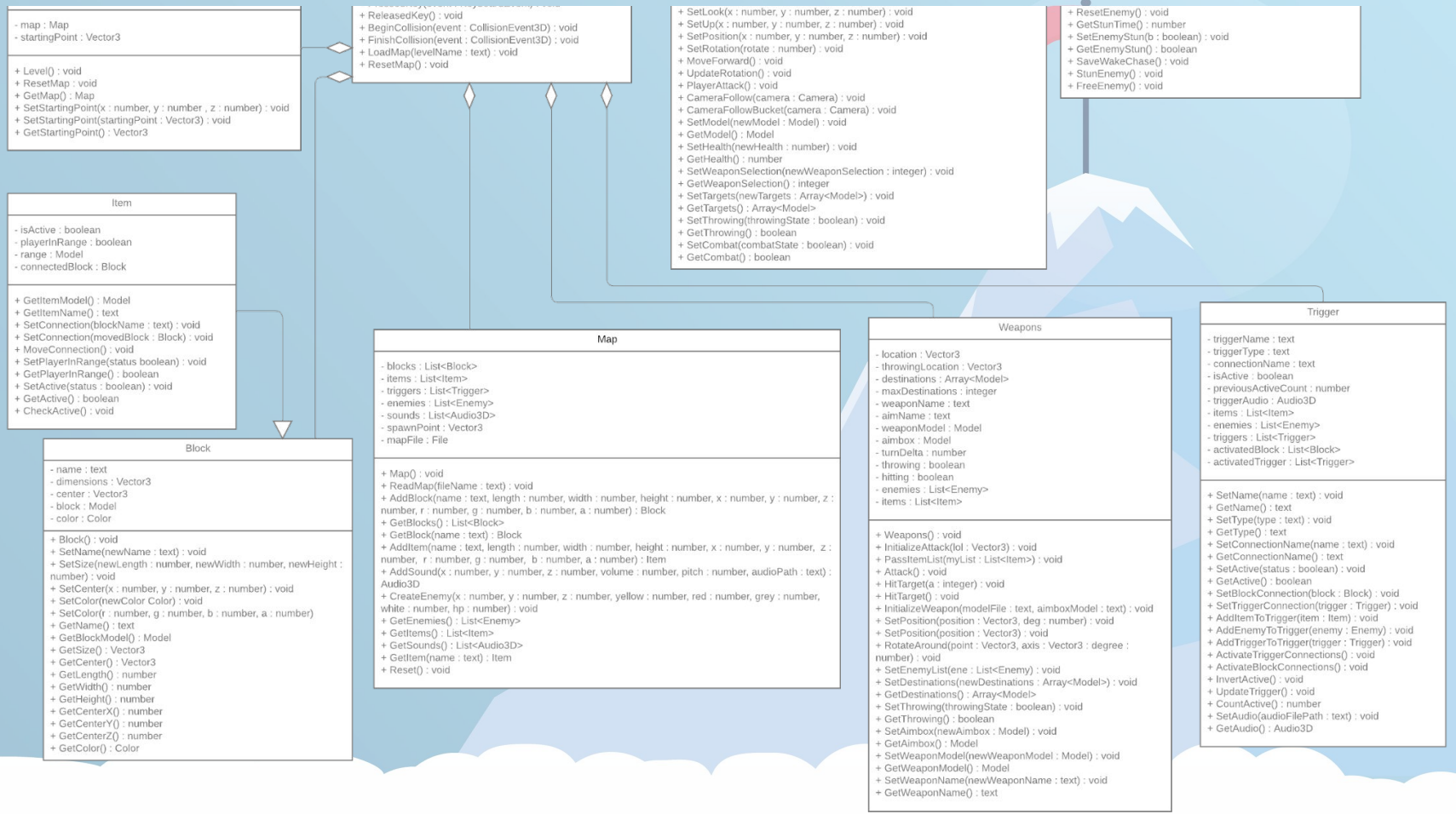
- Operating System - Windows and Mac OS
- Programming Language - Quorum Language
- Software License - Closed Source
- Storyline - A simple yet compelling story for players to uncover

“You must solve these puzzles and survive the challenges to uncover the tragedy of Ento Zero.”

UML Class


Battle
- battlefield : Model
- hail1 : Model
- hail2 : Model
- hailCrash1 : Model
- hailCrash2 : Model
- safeSpot : Model
- soundWall : Model
- playerWeapon : Model
- storySubtitle : Drawable
- attack1 : number
- attack2 : number
- attackNotUse : number
- speed : number
- detectionHide : boolean
- battleReady : boolean
- golemReady : boolean
- timeUp : boolean
- turnOver : boolean
- playerGetHit : boolean
- playerX : number
- playerY : number
- playerZ : number
- playerLocation : number
- leftAlert : Audio3D
- midAlert : Audio3D
- rightAlert : Audio3D
+ Battle() : void
+ CreateBattle(p : LiveBeing, e : Enemy, stage : integer) : void
+ MoveGolemArm() : void
+ PlayerDodge(dodge : number) : void
+ ShootBullet() : void
+ NextWave() : void
+ RestArm() : void
+ PlayDodgeSound() : void
+ FightOver() : void
+ HideDetectionField() : void
+ SetBattlefield(newBattlefield : Model) : void
+ GetBattlefield() : Model
+ SetHail1(newHail1 : Model) : void
+ GetHail1() : Model
+ SetHail2(newHail2 : Model) : void
+ GetHail2() : Model
+ SetHailCrash1(newHailCrash1 : Model) : void
+ GetHailCrash1() : Model
+ SetHailCrash2(newHailCrash2 : Model) : void
+ GetHailCrash2() : Model
+ SetSafeSpot(newSafeSpot : Model) : void
+ GetSafeSpot() : Model
+ SetSoundWall(newSoundWall : Model) : void
+ GetSoundWall() : Model
+ SetPlayerWeapon(newPlayerWeapon : Model) : void
+ GetPlayerWeapon() : Model
+ SetStorySubtitle(newStorySubtitle : Drawable) : void
+ GetStorySubtitle() : Drawable
+ SetBattleReady(battleReadyStatus : boolean) : void
+ GetBattleReady() : boolean
+ SetEnemyWaitTime(newEnemyWaitTime : number) : void
+ GetEnemyWaitTime() : number
+ SetPlayerGetHit(playerGetHitStatus : boolean) : void
+ GetPlayerGetHit() : boolean
Projectile
- model : Model
- fileName : text
- speed : number
- damage : number
- timeActive : number
- maxTime : number
- moveDirection : vector3
+ Projectile() : void
+ MoveForward() : void
+ SpawnProjectile(position: Vector3, look : Vector 3) : void
TitleScreen
- title : Drawable
- arrowX : Drawable
- arrowY : Drawable
- player : Item2D
- sfx1 : Audio
- sfx2 : Audio
- sfx3 : Audio
- menuBGM : Audio
- speech : Speech
- audio1 : File
- audio2 : File
- audio3 : File
- images : File
- scene : File
- menuXSelection : number
- menuYSelection : number
- menuScreen : number
- screenSize : number
- keys : KeyboardEvent
+ TitleScreen() : void
+ Update(seconds : number) : void
+ PressedKey(event : KeyboardEvent) : void
+ ChangeResolution(screenBounds : number) : integer
SaveSystem
- playerData : JavaScriptObjectNotation
- newSave : boolean
- saveEnable : boolean
- saveFile : File
- timer : DateTime
- saveTime : number
+ SaveSystem() : void
+ Save(player : LiveBeing, camera : Camera) : void
+ Load(player : LiveBeing, camera : Camera) : void
+ PlayerSave(player : LiveBeing, camera : Camera) : void
+ PlayerLoad(player : LiveBeing, camera : Camera) : void
+ EnableSaving(save : boolean) : void
+ SetAutosaveTimer(time : number) : void
+ SetSaveTime(newSaveTime : number) : void
+ GetSaveTime() : number
Enemy
- originalX : number
- originalY : number
- originalZ : number
- distanceX : number
- distanceZ : number
- moveDirection : Vector3
- xMovement : number
- yMovement : number
- zMovement : number
- radians : number
- speed : number
- isWake : boolean
- isChase : boolean
- inCombat : boolean
- isStun : boolean
- isWakeBefore : boolean
- isChaseBefore : boolean
- health : number
- maxHealth : number
- timer : DateTime
- model : Model
- caution : Model
- attack : Model
- chase : Model
- proximity : Model
- stunTime : number
+ Enemy() : void
+ UpdateListener(player : LiveBeing)
+ SetEnemyPosition(x : number, y : number, z : number) : void
+ SetBoundry(y : number, r : number, g : number, w : number) : void
+ SetHealth(h : number) : void
+ GetHealth() : number
+ ReduceHealth(h : number) : void
+ ResetHealth() : void
+ IsEnemyChase() : boolean
+ NotifyPlayer() : void
+ StopNotify() : void
+ WarnPlayer() : void
+ BeginChase() : void
+ ChasePlayer(player LiveBeing) : void
+ StopChase() : void
+ ResetEnemy() : void
+ GetStunTime() : number
+ SetEnemyStun(b : boolean) : void
+ GetEnemyStun() : boolean
+ SaveWakeChase() : void
+ StunEnemy() : void
+ FreeEnemy() : void
Main
- title : TitleScreen
- player : LiveBeing
- battle : Battle
- camera : Camera
- monitor : InputMonitor
- keys : KeyboardEvent
- enemy : List<Enemy>
- enemyIt : Iterator<Enemy>
- enemyIndex : integer
- stage : integer
- currentEnemy : Enemy
- saveData : SaveSystem
- block : Block
- level : Level
- blockModel : Model
- blocks : List<Block>
- it : Iterator<Block>
- item : Item
- itemModel : Model
- items : List<Item>
- itemIt : Iterator<Item>
- currentSound : Audio3D
- soundIt : Iterator<Audio3D> : sounds:GetIterator()
- currentSoundPlayingIndex : integer
- hitSound : Audio3D
- speech : Speech
+ Main() : void
+ CreateGame() : void
+ Update(seconds : number) : void
+ PressedKey(event : KeyboardEvent) : void
+ ReleasedKey() : void
+ BeginCollision(event : CollisionEvent3D) : void
+ FinishCollision(event : CollisionEvent3D) : void
+ LoadMap(levelName : text) : void
+ ResetMap() : void
LiveBeing
- model : Model
- front : Model
- bottom : Model
- defaultCameraAngle : Vector3
- defaultCameraOffset : Vector3
- moveDirection : Vector3
- lookDirection : Vector3
- upDirection : Vector3
- xMovement : number
- yMovement : number
- zMovement : number
- detectionHide : boolean
- health : number
- maxHealth : number
- speed : number
- degrees : number
- turning : boolean
- canMove : boolean
- rotation : number
- spear : Weapons
- incombat : boolean
+ LiveBeing() : void
+ LiveBeing(new : Model, look : Vector3, move : Vector3, up : Vector3) : void
+ LiveBeing(new : Model, look : Vector3, move : Vector3, up : Vector3, hp : number, spd : number, de : number) : void
+ GetX() : number
+ GetY() : number
+ GetZ() : number
+ Targeting() : void
+ NewWeapon(weaponName : text, aimName : text) : void
+ SetLook(x : number, y : number, z : number) : void
+ SetUp(x : number, y : number, z : number) : void
+ SetPosition(x : number, y : number, z : number) : void
+ SetRotation(rotate : number) : void
+ MoveForward() : void
+ UpdateRotation() : void
+ PlayerAttack() : void
Level
- map : Map
- startingPoint : Vector3
+ Level() : void
+ ResetMap : void
+ GetMap() : Map

UML Class Continued



User Study

Interviews

- 8 sighted people played our game
- Age range of 16 - 25 years old

Feedback

- Enemy minigame is too easy/gets repetitive
- Lack of volume control was annoying

- Two adult males with cataracts played our game (38 & 44 y/o)

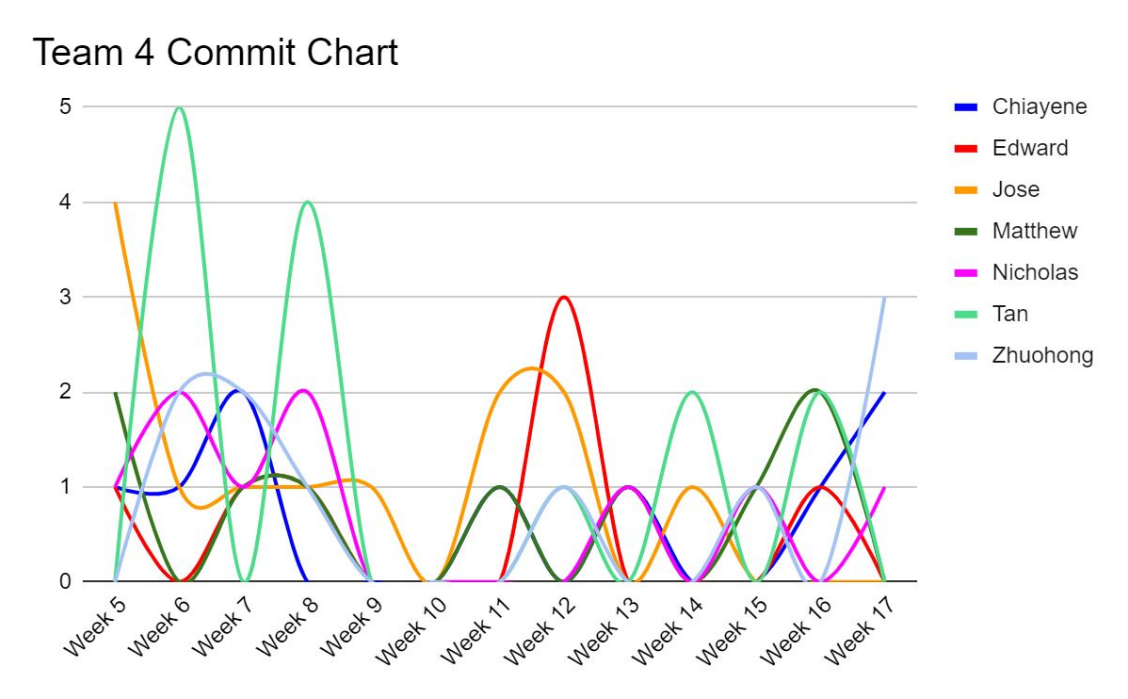
- Audio always sounds like it’s coming from the right
- Provide a controls tutorial to avoid confusion

Changes From Feedback

Additions

- Fast paced combat
- Balanced audio design
- Unique sounds to prevent confusion

Team Commit Counts



Tan - 9

Edward - 8

Matthew - 9

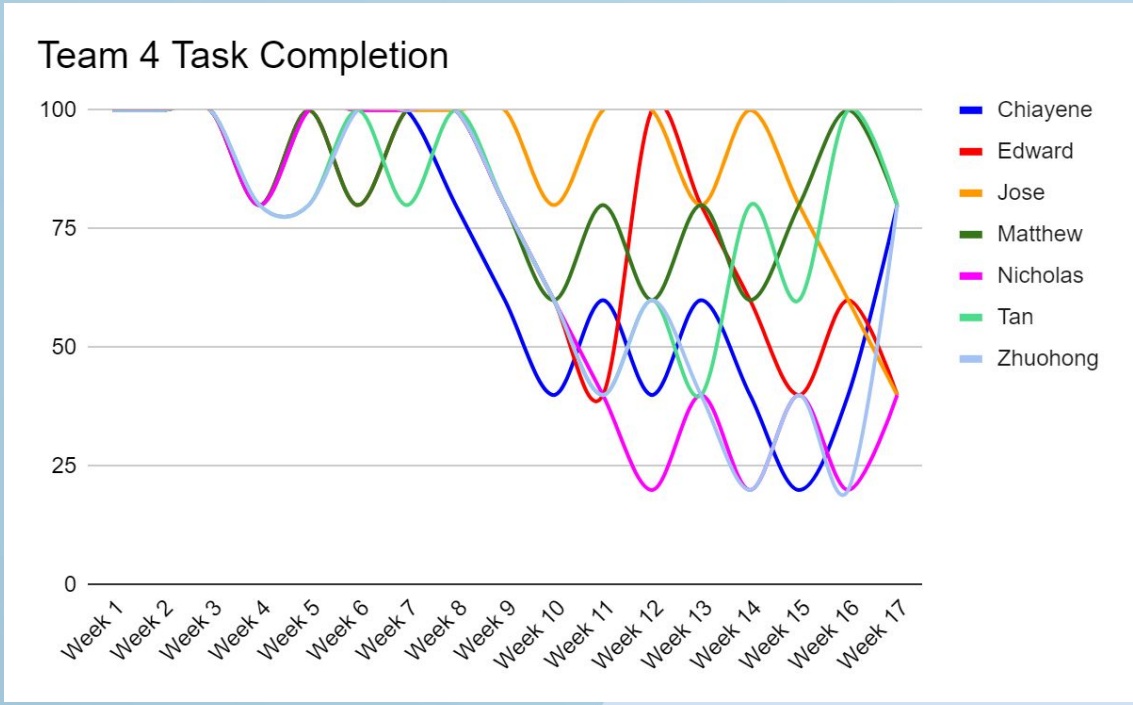
Nicholas - 8

Chiayene - 9

Jose - 9

Zhuohong - 10

Team Tasks & Roles



Core Gameplay

- Nick, Tan, Zhuohong
- Focus: Game Mechanics

UI/Audio Design

- Matthew, Jose
- Focus: UI and 3D Audio

Map Design

- Edward, Chiayene
- Focus: Level Creation

Ento Zero

A Visually Accessible 3D Game

UNLV | HOWARD R. HUGHES COLLEGE OF ENGINEERING

RATIONALE

Modern-day video games struggle to accommodate people who are blind or have a visual impairment. Numerous well-known games from 2019 fail to meet the minimum visual accessibility standards of legible text sizes and optional high-contrast filters [1]. Our team aims to create a 3D adventure game that is inclusive to visually-impaired players with the usage of positional audio and informative sound effects.

DESIGN

Our research on visual accessibility design states that:

- Audio feedback is required for navigation, current player status, and nearby entities. [2]
- Line-of-sight tools alone fail to communicate environment layouts and scenery to blind players. [3]

With these findings, our team concluded on utilizing immersive audio design for conveying information. We constructed our desktop game with Quorum Studio.

FEATURES

- Positional Audio Spatialization
- High-Contrast Graphics
- Overworld Exploration & Puzzle-Solving
- Weapons & Combat Mechanic

GAMEPLAY

Main Menu

Menu System

An accessible menu that has built in screen reader compatibility and screen resolution options.

Environmental Sound-Effects

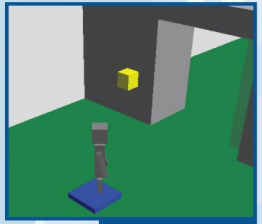
Determine your surroundings by listening for enemy alerts and pings.

Puzzle-Solving

Solve puzzles by pressing buttons, pressure plates, or defeating enemies to unlock new areas.

Enemy Encounters

Engage in combat or avoid enemies by stunning them with a spear.



Overworld

Combat

Fast-Paced Gameplay

Avoid enemy attacks by moving to a safe position and counterattack after dodging correctly.

Distinct Audio Cues

Listen carefully for one of three sound effects that represent the upcoming safe position: dodge left, stay center, and dodge right.

FEEDBACK

We interviewed 8 sighted individuals between the ages 16-25. And have adapted to feedback by adding:

+ Fast paced combat
+ Balanced audio design
+ Gameplay guide to prevent confusion

CONCLUSION

Our team has completed a demo that will let players explore the features in Ento Zero.

"Listen, solve, and survive the challenges to uncover the tragedy of Ento Zero"

REFERENCES

EDWARD HEAR | NICHOLAS HEERDT | ZHUOHONG LEI | CHIAYENE MAGBANUA | TAN PHAN | MATTHEW RUEST | JOSE SISON