Technical Manual For Reference
(https://www.studytonight.com/3d-game-engineering-with-unity/tdd-and-gdd)

Technical Manual

Technical Manual for *StarBeasts* by Team Russian Blue for CMSC447

Introduction

Starbeasts is a single-player, outer space-themed, Pokémon-inspired RPG. The game's main feature is its combat. Keeping on theme, the battle is turn-based and done with a party of Aliens. This party fights various enemies and groups of enemies that may differ depending on which area (level) the player is in.

The player starts with two Aliens in their party, but they will be able to procure more by Catching Aliens after passing the level or random encounter or by Purchasing new Aliens from stores.

Stores also contain items for purchase which can be used on the player's aliens to increase their stats or for advantages in battle.

Development Libraries and Languages

The heart of the game is built with the Phaser[1] game framework. Phaser is based on JavaScript and includes built-in functions through the library to help aid game development. The game also utilizes the Python module Flask as a web framework. SQLite was chosen for the database engine, which was needed to store saved game information. Finally, the game utilizes HTML5 and Bootstrap CSS for styling and web-specific design features.

Ultimately, Phaser was chosen as it seemed to have the functionality needed to enable the game. It also had easy-to-use documentation and examples for the team to use while learning. Flask and SQLite were chosen as the team was already familiar with their use, and they were compatible with the Phaser game framework.

[1] https://phaser.io

Development Schedule

The project ran on a Scrum schedule with 3 major sprints, each lasting a month. Sprint One was dedicated to formulating the game ideas and creating documentation for easier implementation. Sprint Two was largely focused on writing the majority of the code for the game. The final sprint, sprint Three, was for ironing out any remaining game logic and testing the written code.

Software Design Documents

FlowCharts

The menu diagram is a high-level flowchart of how to load and engage with the game [Figure 1].

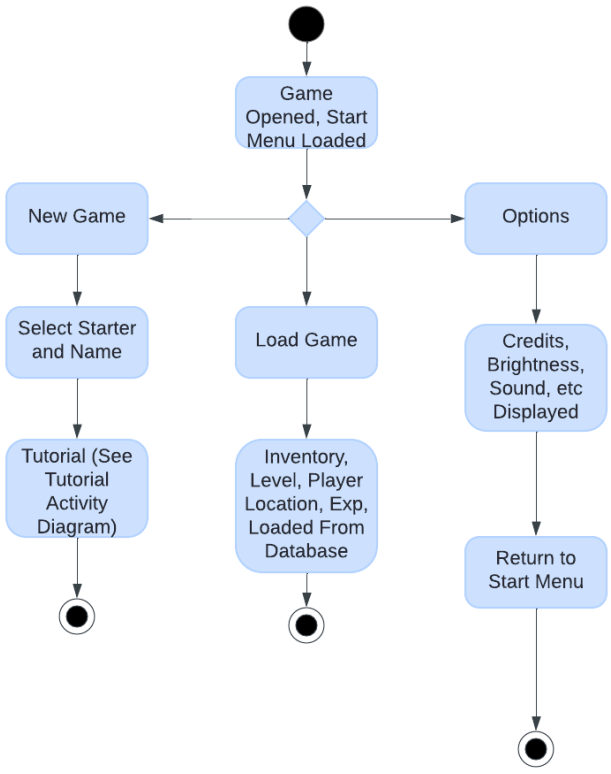


Figure 1: Menu Diagram

The gameplay diagram outlines how the player plays the game once loaded [Figure 2].

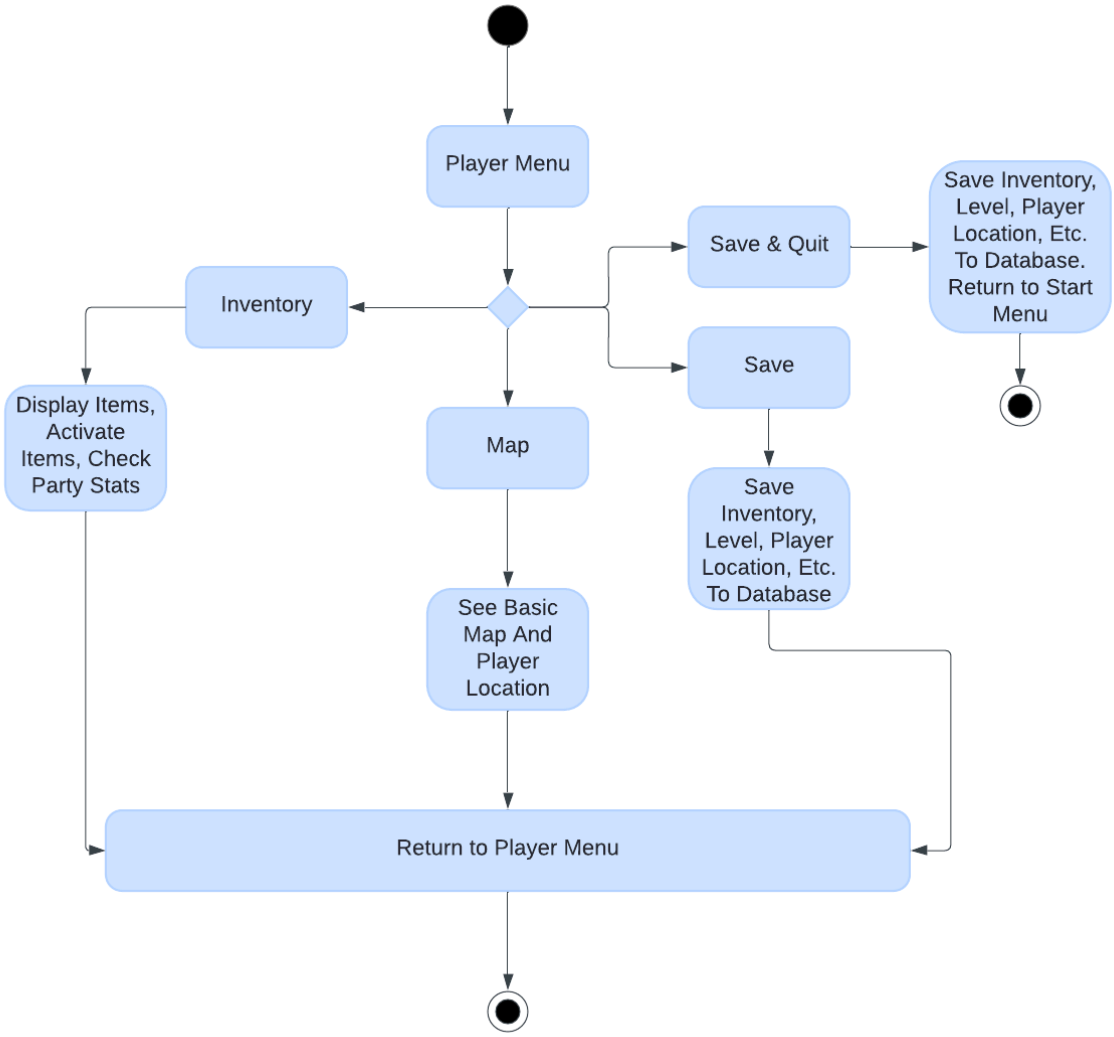


Figure 2: Gameplay Diagram

There is an optional tutorial for new players. To access, it must be selected from the new game menu. The tutorial diagram is a flowchart of how to access the tutorial [Figure 3].

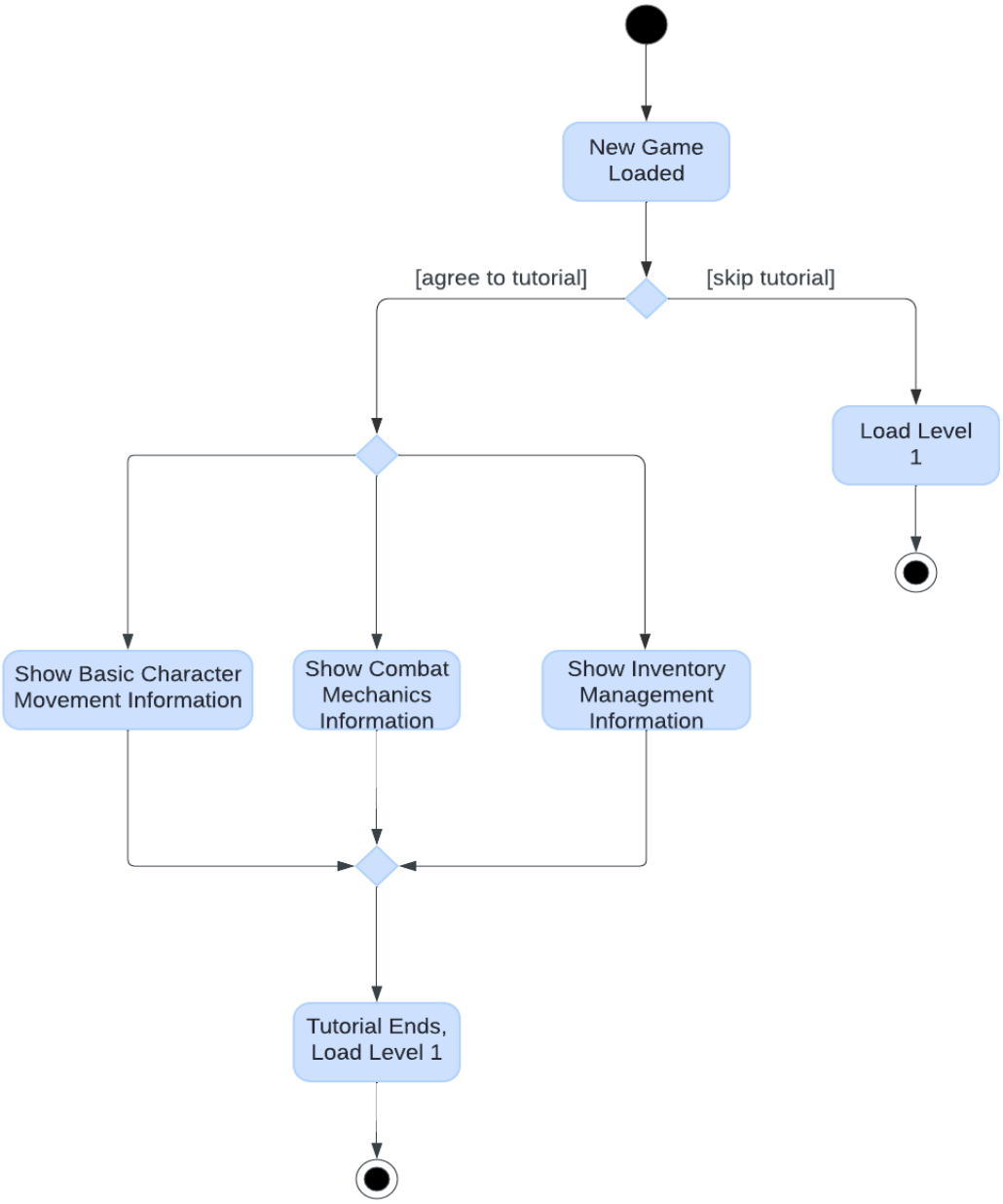
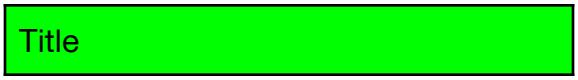
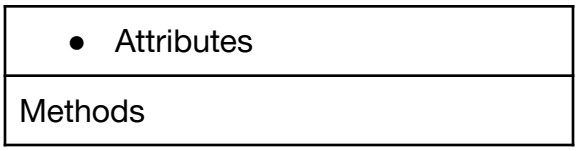
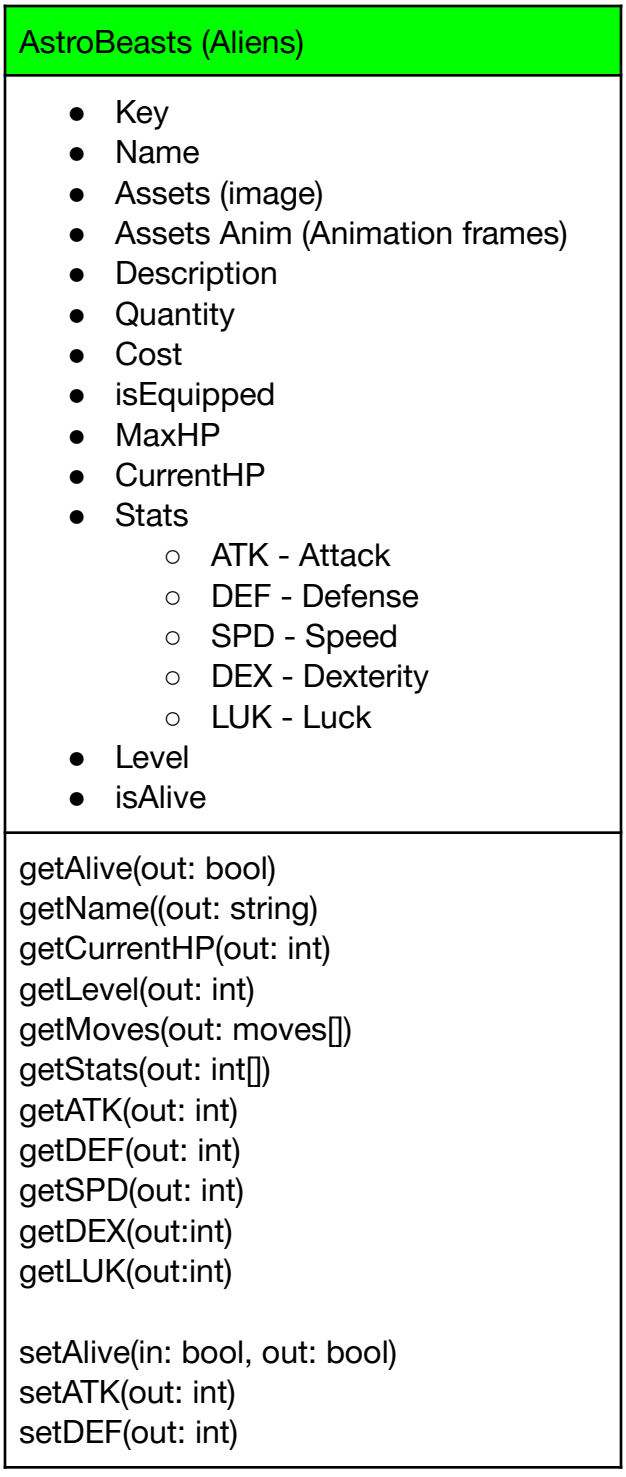


Figure 3: Tutorial Diagram

Class Diagrams




• Attributes
Methods


AstroBeasts (Aliens)
• Key
• Name
• Assets (image)
• Assets Anim (Animation frames)
• Description
• Quantity
• Cost
• isEquipped
• MaxHP
• CurrentHP
• Stats
○ ATK - Attack
○ DEF - Defense
○ SPD - Speed
○ DEX - Dexterity
○ LUK - Luck
• Level
• isAlive
getAlive(out: bool)
getName((out: string)
getCurrentHP(out: int)
getLevel(out: int)
getMoves(out: moves[])
getStats(out: int[])
getATK(out: int)
getDEF(out: int)
getSPD(out: int)
getDEX(out:int)
getLUK(out:int)
setAlive(in: bool, out: bool)
setATK(out: int)
setDEF(out: int)

setSPD(out: int)
setDEX(out:int)
setLUK(out:int)

takeDamage(in: int)
gainExp(in: int)
levelUp(out: increase stats)

Enemy

- Name
- Assets (image)
- Assets Anim (Animation frames)
- MaxHP
- CurrentHP
- Stats
    - ATK - Attack
    - DEF - Defense
    - SPD - Speed
    - DEX - Dexterity
    - LUK - Luck
- Moves
- Level
- isAlive

getAlive(out: bool)
getName((out: string)
getCurrentHP(out: int)
getLevel(out: int)
getMoves(out: moves[])
getStats(out: int[])
getATK(out: int)
getDEF(out: int)
getSPD(out: int)
getDEX(out:int)
getLUK(out:int)
setAlive(in: bool, out: bool)
takeDamage(in: int)

Shop Items

- Key
- Name
- Description
- Quantity
- HP
- ATK
- DEF
- SPD
- DEX
- LUK
- isSelected

- Effect

Inventory

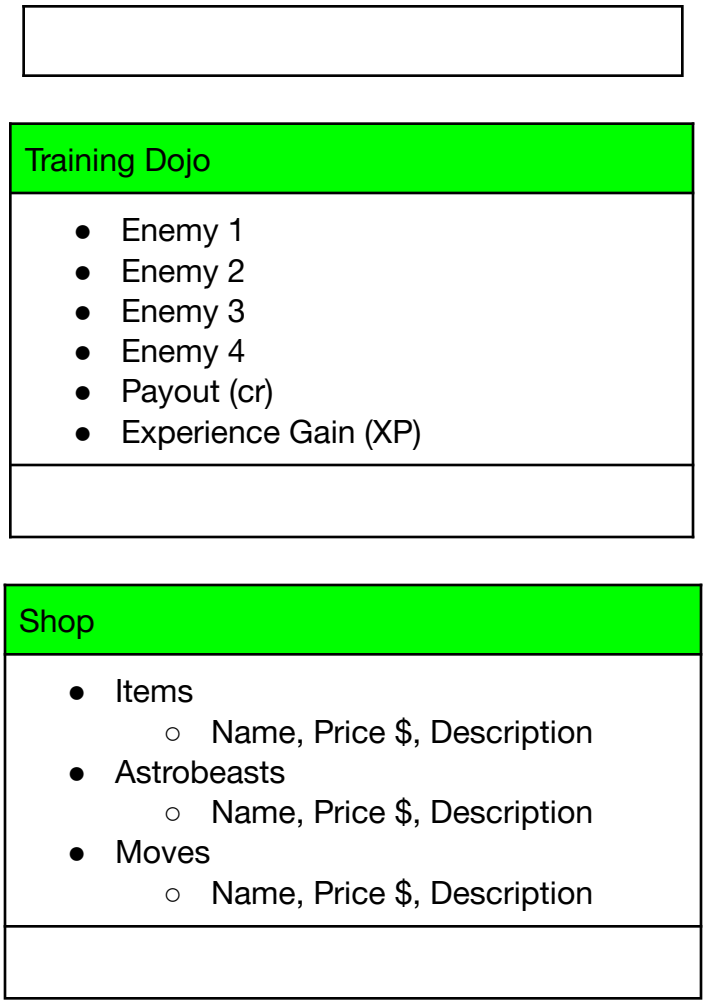
- Items
- Currency

Boss / Enemy NPC

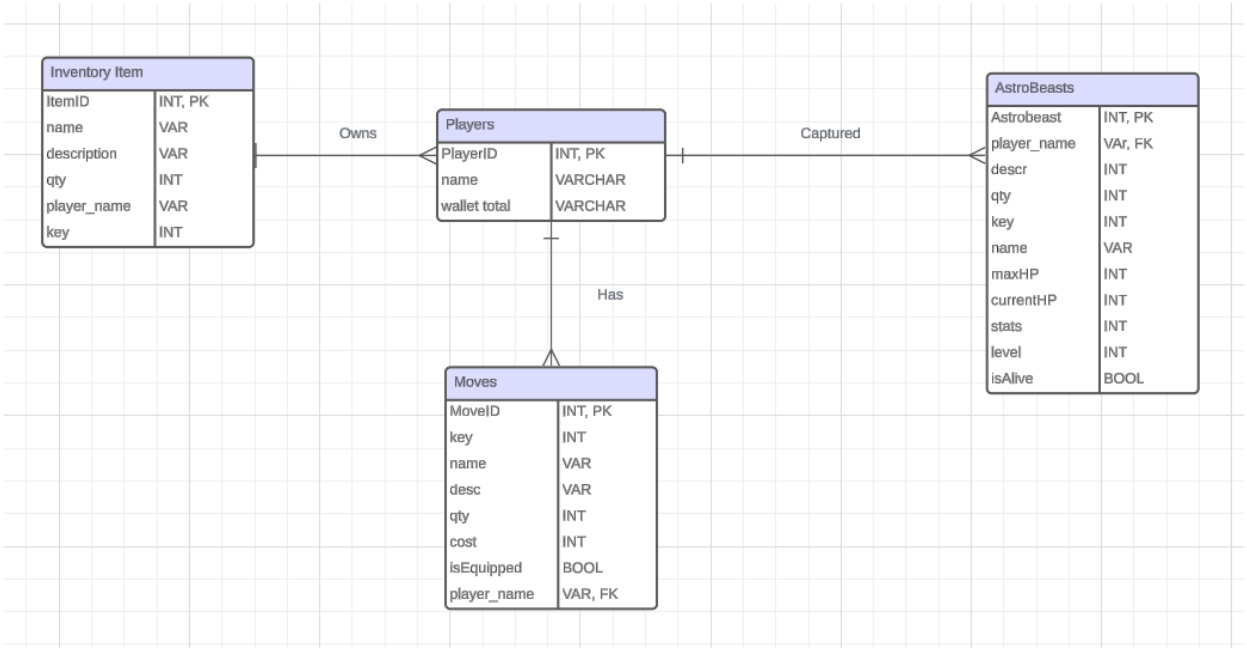
- Name
- AstroBeasts
    - Slots 1-4

Save Slots

- Save Slot Name (SSN) : 5 chars long
- Inventory
- Arena tournament layout
- Defeated Bosses


Training Dojo
Enemy 1
Enemy 2
Enemy 3
Enemy 4
Payout (cr)
Experience Gain (XP)
Shop
Items
Name, Price $, Description
Astrobeasts
Name, Price $, Description
Moves
Name, Price $, Description

E-R Diagram



Game Assets

Sprites (AstroBeasts)

The main assets for the game are the battling Astrobeasts. There are multiple types of Astrobeasts in the game, listed in Table 1.

Table 1: List of AstroBeasts

Types of Astrobeasts		
Zallo	Icell	Grath
Hotu	Scourge	Ragnex
Shamrock	Aesun	Strikoh
Tarkeel	Tyboar	Malgrun
Skol	Scourge	Ruinn
Arquam		

Each Astrobeast has inherent Stats. There are multiple categories of stats, with each Astrobeast inherently containing particular groups as listed in Table 2. Table 3 displays a complete accounting of each Astrobeat and their base (Level 1) stats:

Table 2: List of AstroBeast Stats

Stat Abbreviation	Stat Full Name	What the Stat does
ATK	Attack	Intensity of Attack
DEF	Defense	Mitigates damage from opponent attack
SPD	Speed	Influences turn order in battle
DEX	Dexterity	Influences the chance of attack missing player
LUK	Luck	Influences the chance of attack missing player
ELM	Element	Category of Astrobeast which changes inherent strengths and weaknesses
EXP	Experience	Gained in battle, contributes toward Astrobeast's level

Table 3: List of AstroBeasts and Base (Level 1) Stats

Name	Rarity	Class	ATK	DEF	DEX	SPD	LUK	HP	St/Res
Skol	Common	Balanced	300	250	300	300	250	1,000	
Tarkeel	Common	Assassin	194	128	448	500	130	1,000	
Arquam	Common	Tank	266	470	198	312	154	1,000	
Shamrock	Common	DPS	358	120	218	246	458	1,000	
Hotu	Common	DPS	476	342	144	226	212	1,000	
Zallo	Common	Tank	434	470	122	102	272	1,000	
Icell	Rare	Tank	410	648	442	510	310	2,000	
Ragnex	Rare	Balanced	520	514	450	478	438	2,000	

Strikoh	Rare	DPS	632	408	474	468	418	2,000	
Scourge	Rare	DPS	550	498	396	416	540	2,000	
Aesun	Legendary	Assassin	864	820	1086	1018	612	3,000	
Tyboar	Legendary	Tank	832	1088	656	1,046	808	3,000	
All Wrath	Legendary	DPS	1,096	982	504	1,040	778	3,000	
Malgrun	Galactic	DPS	1,980	1,380	1,692	1,184	1,064	4,000	
Ruinn	Galactic	Assassin	1,114	1,556	1,382	1,260	1,988	4,000	

Items

Items are consumables that can be purchased in the Shop and can provide advantages to the AstroBeasts in the player's party when used. They are all single use items, and are deleted from the player's available inventory when used in battle.

Offensive Gadgets

- Titanium Claws: Increased ATK by 10 for 1 ally AstroBeast
- Sonic Boosters: Increased SPD by 10 for 1 ally AstroBeast
- Hardlight Shell: increases DEF by 10 for 1 ally AstroBeast
- Sim Beacon: Increased DEX, LUK by 10 for 1 ally AstroBeast

Defensive Gadgets

- Photon Shield: Increased DEX, LUK by 10 for 1 ally AstroBeast
- Photon Armor: block 1 attack against an ally Astrobeast. Lasts until hit.
- Photon Barrier: block 2 attacks against an ally Astrobeast. Lasts until hit 2.
- Radiation: Removes elemental weakness for 1 turn.
- First Aid Tape: Heals an injured Astrobeast for 25% hp.
- Biofoam: Heals an injured Astrobeast for 50% hp.
- MedDrone: Heals an injured Astrobeast for 100% hp.

Snacks

- CosmoCookies: Heals an injured Astrobeast for 15 hp.
- AstroAde: Heals an injured Astrobeast for 30 hp.
- SequidSamdo: Heals an injured Astrobeast for 60 hp.

Revives

- BactaDrone: A defeated Astrobeast is revived to 40% hp.

Backgrounds

There are three combat backgrounds, a shop background, and a number of loading/transition screens. The backgrounds mainly came from a generative AI tool, Perchance[2], although other online content generation tools were also used and customized to meet the game design requirements.

Attacks

The types of attacks that are available are **Damage (Table 4)** and **Elemental (Table 5)**

Table 4: List of Damage Category Attacks

Damage Attacks		
Slash	Headbutt	Punch
Cross-Slash	Scratch	Elbow
Chomp	Body Slam	Knee
Pounce	Throw	Strike
Tackle	Ram	

Table 5: List of Elemental Category Attacks

Elemental Attacks in increasing Strength Order

[2] https://perchance.org/ai-character-generator

Fire • Ignite • Fireball • Flamethrower • Incinerate · Solar Flare	Water • Splash • Hydro Kai • Torrentus • Tsunami • Quasar Reservoir	Rock • Harden • Rocksteady • Seismic Slam • Earthquake • Meteoric Rise
Scorch: • Spark • Shock Strafe • Lighting Jolt • Thunderspear • Cosmic Storm	Gravity • Repulsor • State Warper • Mass Surge • Gravitic Flux • Black Hole	

Game Logic

The actual combat mechanics are loosely based on Pokemon. The player has a party of up to four AstroBeasts which battle against arrays of enemies. The turn-based combat continues until one side is defeated. To determine which side starts first, the player or the enemies, we calculate the average speed (SPD) stat of both sides's AstroBeasts, and whichever side is higher starts first. If every AstroBeast in the current party (enemy party or player party) has ended their turn, and there is at least one living enemy, the other party begins their turn. This gameplay continues until one side is defeated. If the Player wins, they are rewarded with Credits and XP. If they lose, they lose some Credits and respawn at the closest save.

Four basic combat operations are offered: FIGHT, FLEE, ITEM, and SCAN. These are described in more detail below, along with a flowchart [Figure 4].

- FIGHT— AstroBeast performs physical and elemental attacks
- FLEE— The player's party attempts to run away and end combat
- ITEM— The player uses an item on an AstroBeast which consumes/applies the item
- SCAN—Get info on enemy resistances/weaknesses (this won't end turn).

FIGHT

When the battle starts, the player's Party of AstroBeasts (max 4) will be set against a randomly generated party of enemies (max 4.) Combat is in the form of a turn-based battle that occurs in rounds until one party is completely defeated.

- When this option is selected, a list of the AstroBeast's learned attacks will appear in the GUI as well as the basic stats of that attack.
- The player will be prompted to choose an attack.
- When an attack is selected, the player will then be prompted to select a target enemy to use that attack on.
- Once the attack and target have been chosen, the system will calculate if the attack landed based on the attacker's dexterity (DEX) and luck (LUK) stats. **If not, this AstroBeast's turn ends.**
- If the attack lands successfully, the system will calculate the raw damage of the attack by using the AstroBeast's attack (ATK) stat, the attack's damage **(DMG)** stat, and a random number.
- The system will calculate the amount of damage that will be mitigated according to the target's defense (DEF) stat and a random number.
- The total damage is given by taking the difference between the raw attack and mitigated damage. This total damage is displayed on the screen.
- This total is then deducted from the target's health pool. If this deduction drops them to or below zero Health Points, they are dead
- **The Astrobeast's turn ends.**

FLEE

- Selecting this option will begin an attempt to forfeit the battle.
- This is calculated by random chance. The player has a 20% chance to flee the battle.
- Upon successful fleeing, **the battle ends.**
- If fleeing is unsuccessful, **the AstroBeast's turn ends and the battle continues**.

ITEM

- Selecting this option will bring up the player's inventory to the GUI.
- Once an item to give to the AstroBeast has been selected, the item's effect will be performed.
- The item is consumed and removed from the inventory.
- **The AstroBeast's turn ends.**

SCAN

- Selecting this option will allow the player to scan an enemy combatant
- Scanning will display a description of the combatant and their remaining health
- This does not end the AstroBeast's turn

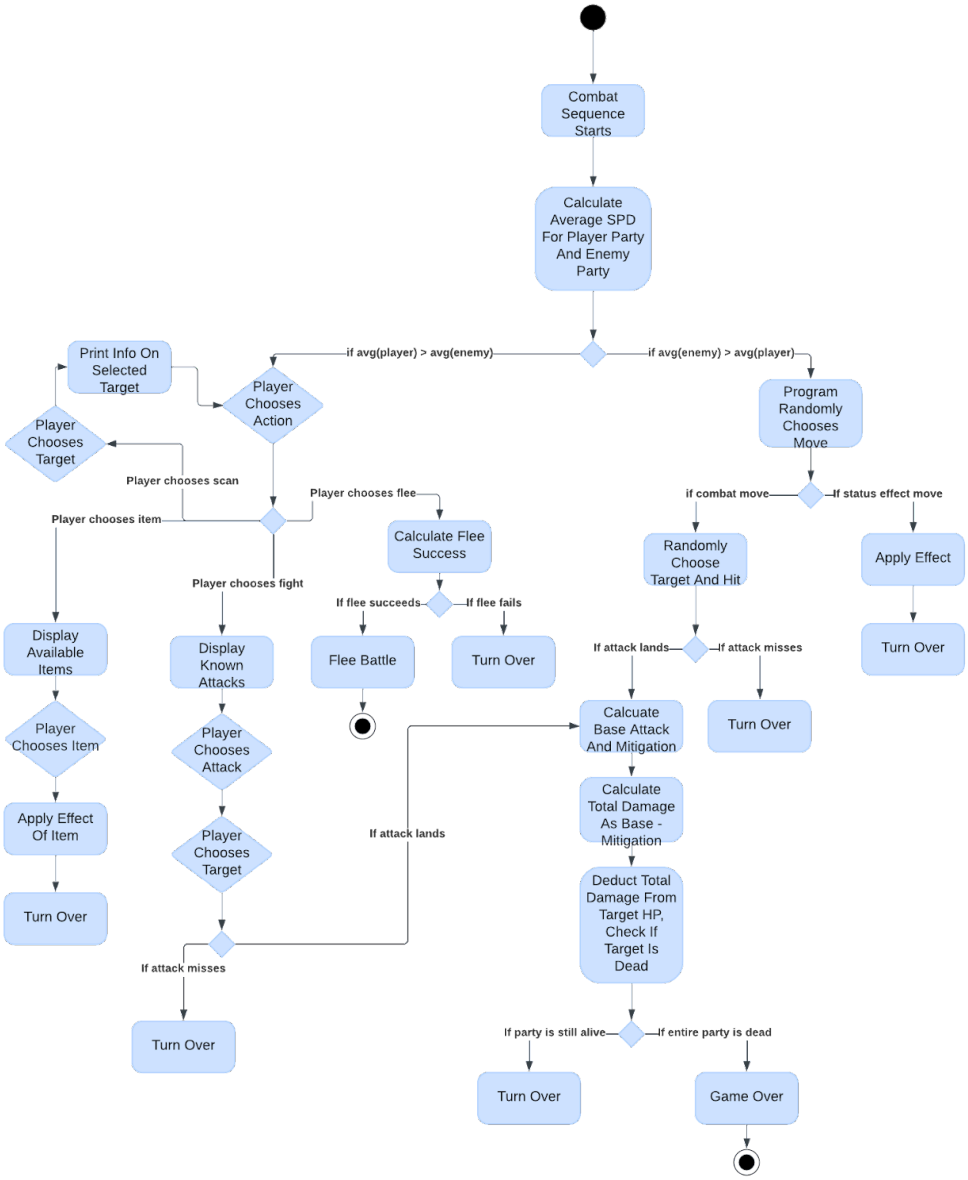


Figure 4: Combat Flowchart

Networking

The game is a single-player game. The application is currently hosted on Github Pages.

Hardware/Software Requirements

The game is compatible with MacOS and PC. The game has been tested on Chrome, Edge, Safari. Generally, any modern operating system will support the game.

API Information

The file routes.py handles all of our routing and is home to the application's endpoints. It handles the various HTTP requests and responses for our application. As mentioned previously, our application uses Flask for our backend web framework and SQLite for the database. The endpoints are listed below:

/ (Root Endpoint)

The methods used here are both GET and POST. This endpoint marks the start screen of our application. There is no request for GET. The response is a rendering of the game.html template, which points to our Phaser code.

/save_game

The method used here is POST. This endpoint saves the current game state in the format needed for the game's database. The request includes the game state variable that saves all important game state information into one large dictionary, including the player's name, wallet, score, and inventory. The response would be either a success or an error.

/check_name

The method used here is POST. This endpoint checks if a player's name exists in the game's database. If so, their data is retrieved and loaded in. The request is the player's name, and a

response would either be false if they don’t exist, or true and all of their information retrieved from the database.

/get_high_scores

The method used here is GET. This endpoint retrieves the top 5 player scores. A response would be the names and scores of those players.

/submit_scores

The method used here is POST. This endpoint submits the top 5 player scores to Dr. Allgood’s public API. The request would be the specific structure below:


```
{
   "data": [
     {
       "Group": "Russian-Blue",
       "Title": "Top 5 Scores",
       "AJ": 0,
       "aamil": 0,
       "bob": 0,
       "josh": 0,
       "leann": 0
     }
   ]
 }
```

The response would either be a success or an error from the API.