Star Blade

Ian Beard
ian.beard@bellevuecollege.edu
Michael Lablanc
michael.lablanc@bellevuecollege.edu
Yasir Egal
yasir.egal@bellevuecollege.edu
John Sablaon
john.sablaon@bellevuecollege.edu

Scope

We want to make a web based single player turn based RPG using HTML5 Canvas. The scope of the full project is potentially quite large. While perhaps not as massive as some notorious "90 hour RPGs," this project could still be quite the undertaking

For the purposes of building a prototype, however, development will initially be limited to letting users play through a single battle.

Which would play out as selecting a group of characters. The computer generates a group of enemies. Then a character and enemies take turns, selecting actions to eventually defeat the other group until either the player defeats all the enemies and wins or loses when the enemy defeats all of the player's.

A turn for a player's character will start by them generating action points, which then will allow them to do one or more actions, such as attack or heal, until they are out of action points. At which point their turn is over, and the next character or enemy will get their turn.

Business Narrative

General Specs

- The game will be a single player turn based RPG. The prototype will mainly focus on the combat system.

- The game will use HTML5 Canvas for the visual elements. Pre-built game engines may already exist for HTML5, this will be researched early in development.

- The game will use Node.js and React (Subject to change)

- The database will use MySQL.

- The prototype will allow users to select both the player characters they want to use, and the enemies they would like to fight.

Combat System

Players will win after incapacitating all opponents. They will lose if all of their characters are incapacitated

Characters (playable or otherwise) have 6 main stats:

- Strength (determines physical attack damage)
- Willpower (determines strength of magical attacks and increases resistance to mental status ailments)
- Dexterity (determines accuracy of physical attacks)
- Focus (determines accuracy of magical attacks)
- Defense (reduces damage of incoming attacks and increases resistance to physical status ailments)
- Agility (reduces accuracy of incoming attacks)

Weapons will have a block rate and attack power assigned to them. Attacks will have damage values, accuracy ratings, and critical hit rates assigned to them.

Characters have 4 resources to manage during battle:

- Health (Reduced by taking damage from attacks, gained by healing. Characters are incapacitated when their health reaches 0.)
- Action Points (Characters gain a number of action points each turn, up to a certain limit. These points are spent to perform actions, like attacking or casting spells. Characters may take as many actions as they like per turn so long as they have enough points to spend <Limited to one attack per turn>.)
- Essence (Used to cast spells. Rather than a number of points being spent, Essence is accumulated by a character when they cast spells. If they continue to cast spells after they have reached their maximum amount of essence, they will take damage.)
- Tension (Represents a characters morale, measured as a percentage value. Tension is gained by scoring critical hits or from support spells, and is lost by missing, taking damage, or from certain

enemy skills. High Tension increases accuracy and grants access to powerful actions, low tension lowers accuracy and prevents some actions from being performed. Tension also effects how resistant a character is to mental status ailments.)

Enemies will target characters based on their level of aggression (Aggro) towards them.

- Aggro towards each player controlled character is tracked separately by each enemy.
- Aggro towards a character is gained when they deal damage (only gained by the enemy taking damage), cast spells, or when they use abilities that explicitly state that it increase aggro.
- Aggro is lost when a character takes damage (only the aggro towards the character taking damage) or uses an ability that explicitly states that it decreases aggro.
- Players will be able to arrange their characters into a front and back battle line. Characters on the back line will gain aggro at a reduced rate, but will be unable to perform melee attacks.

User Stories/Test Plan

1. As a player, I need to be able to see which character and enemies I can select, so I can establish characters and enemies before starting the game.

   Test: List of characters and list of enemies displayed on screen

2. As a player, I need to be able to select which characters and enemies I want to play with and against, so I can start the game.

   Test: Characters/enemies can be added to list of selected characters/enemies

3. As a player, I need to be able to start the game after selecting the characters, so the game enters the gameplay.

   Test: The screen displays the battle field and battle menu

4. As a player, I need to be able to view all characters and enemies on the screen, so I know that the game has started.

   Test: The battle starts will all selected characters/enemies present on the screen

5. As a player, I need a visual indicator on the screen to reference character, or enemies turn so I know whose turn it is.

   Test: Visual indicator for who's turn it is should be on screen

6. As a player, I need to be able to pause and resume the game, so that the game will stop and continue as requested.

   Test: Pause menu can be opened and closed at any time

7. As a player, and when the game is paused, I need to be able to restart the game at any point so that the current game will end and start a new game with the same characters starting with the initial state.

   Test: The battlefield is reset to its initial state, with all selected characters and enemies on screen

8. As a player, I need to be able to exit the game at any point, so that I can select new characters and start the game over.

   Test: The player is returned to the start screen

9. As a player, and when the game is paused, I need to be able to view the controls so I can make a selection

   Test: The controls are displayed on screen

10. As a player, and when the game is paused, I need to be able to review the tutorial of the game mechanics.

    Test: The Tutorial can be opened and closed via the pause menu while playing the game

11. As a player, I need to be able to access the tutorial from the selection screen, so that I can review the game mechanics.

    Test: The tutorial can be opened and closed from the select screen

12. As a player, I need to be able to see the health of all characters and enemies on the screen, so I can strategize for my next play.

    Test: A health bar can be seen below each character/enemy on screen

13. As a player, I need to be able to see which enemies are targeting which character, so I can anticipate the damage and prepare my next steps.

    Test: A visual indicator for enemy aggro can be seen next to each enemy

14. As a player, I need to be able to view the characters action to perform, so that I can select one from the menu to execute.

    Test: Available actions can be seen and selected via the battle menu

15. As a player, I need to be able to view the status of a character, so I can decide what my next move will be.

    Test: Characters can be selected from the battle menu, which displays a popup showing their current status

16. As a player, I need to be able to select a valid target, so my character can perform the action selected to the target.

    Test: Players should be able to see a visual indication of who they are targeting with their abilities

17. As a player, I need to be able to select when to end my turn based on available action points, so I can end my turn strategically.

    Test: Performing a single action should not end the turn

18. As a player, I need to be able to view the status and resources of an active character, so I can decide what actions to perform in my next play.

    Test: The “resources” of the active character should be seen next to the battle menu

19. As a player, I need to be notified with a victory screen when all enemies are defeated, so I know the game is over.

    Test: A victory screen must be shown after winning a battle

20. As a player, I need to be notified when all of my characters are defeated with a defeat screen, so I know the game is over.

    Test: A defeat screen must be shown after losing a battle

21. As a player, I need to be able to select to play the game again when it ends, so that the game will start over from the beginning.

    Test: Players must have an option to return to the select screen after winning or losing.

Technical Specifications

1. When the player clicks the start game button, the system will change the page and display a view of all characters and enemies for the game.

2.When the player is finished selecting the characters and enemies for the game, the system will continue and start the gameplay. If no characters or enemies were selected, the system will prompt the user to make selection.

3.When the player is finished selecting the characters and enemies for the game, the system will allow the user to start the game by and display the battlefield and menu.

4.When the gameplay begins, the system will display all characters and enemies on the screen with the appropriate menu to interact with the game.

5.When the game has begun, the system will display clear visual indicators that will highlight the current turn in the round.

6.During the gameplay, the system will provide an option for the player to pause the game by clicking the pause button and resume the game by clicking continue or resume.

7.When the game is paused, the system will provide an option to restart the game with the same characters in the initial state by clicking the restart button.

8. When the game is paused, if the player clicks on exit, the system will change the page to the main menu. Then if the player selects a new game, the previous game state will be reset to allow for a new selection of characters and start a new game.

9. When the player hits the pause button, an image of the controls will display, alongside a resume and exit.

10. When the game is paused. Display a button that will display a description of the mechanics of the game.

11. When the player clicks on the tutorial button, before the character selector, display changes to the tutorial. After the character selection page, this option is removed.

12. The gameplay page initiates with all characters and enemies displaying their current health. As unit health changes, the health displays changes accordingly.

13. As the game progresses, enemies will display their aggro targets. As the value changes internally from attack actions, the target display changes based on the highest aggro value.

14. When it is a character's turn, their action list will display. Once their turn is up, the action will change to the characters action list.

15. The player will check if the character has a valid status in order to choose what move they would like their character to do.

16. The player has to set a specific target so the character knows where to perform the abilities that they were told to perform.

17. A player can take a variety of turns in one go as long as the character has enough Action Points(APs), so that the user can play the game in a smarter way rather than the game moving on after one ability

18. Next to the battle menu there should be a Resources tab where the player will be able to see the status of your character and the abilities to choose from.

19. The player will be notified that they have won the game and that it’s over with a winning screen.

20. The player will be notified that they lost the game and that the game is over by a losing screen being displayed.

21. The player will be able to play again if they would like to restart a match.

Use Case Scenarios

Player Using Battle Menu	Check Character Stats
	See Active Character Resources
	See Active Character's Available Actions
	See Description/Cost of Available Actions
	Select Action from List of Available Actions
	Deselect Action Before Choosing Target

External Interfaces

This system will use an external database to store information about characters and enemies. When the game is loaded, players will be provided with a list of characters and enemies to choose from that is populated by this database.

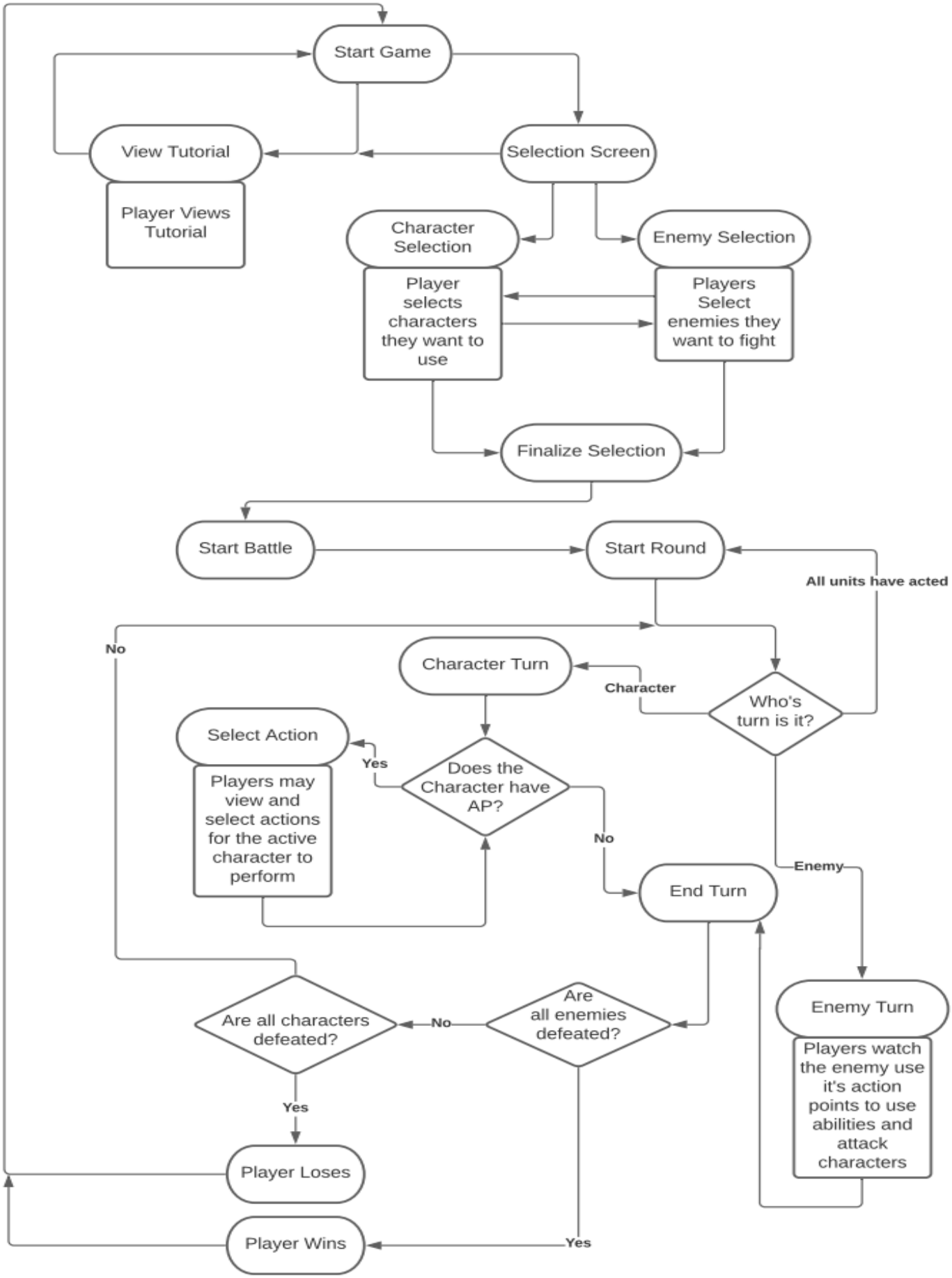
API Calls

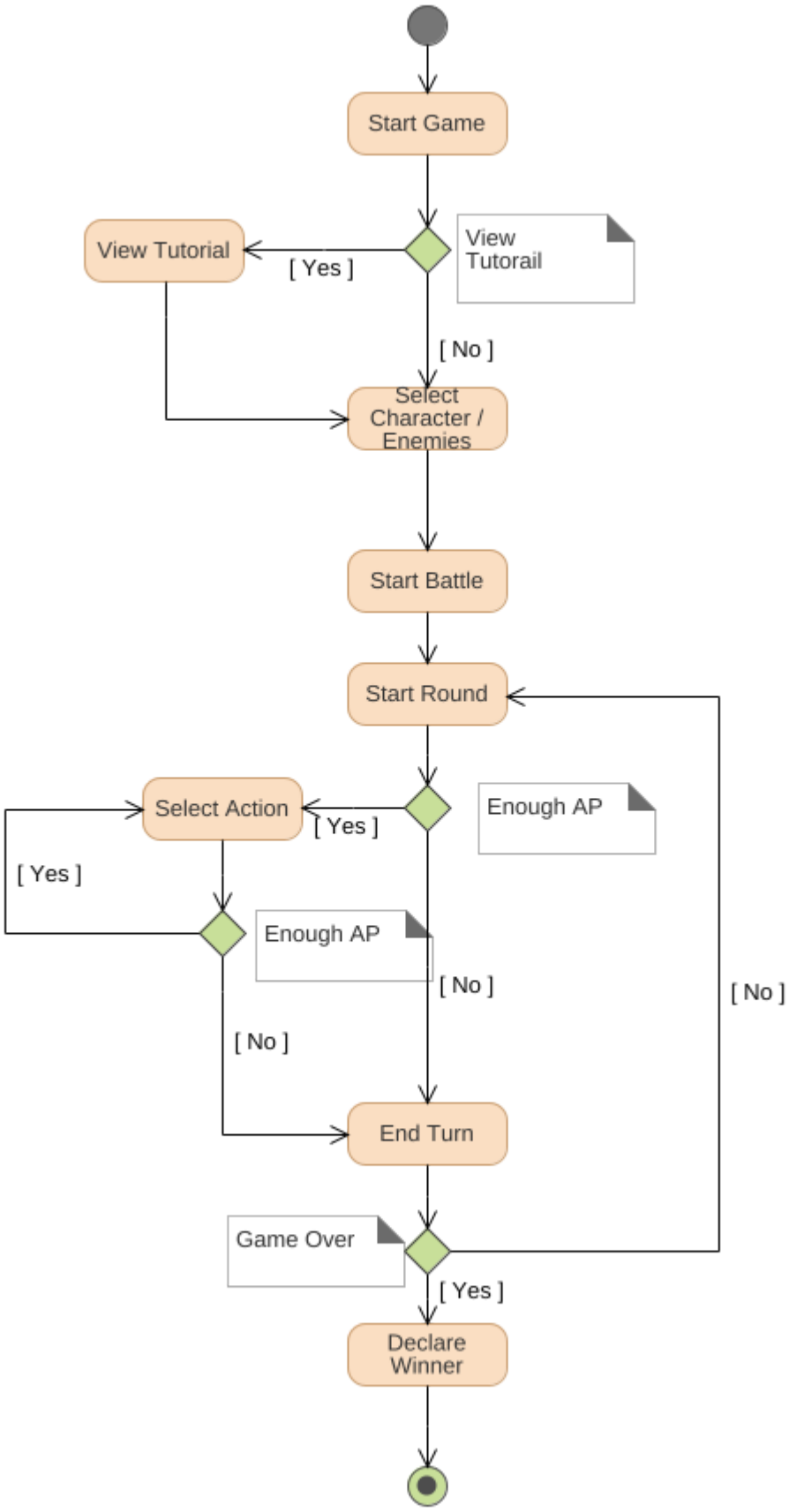
There will be one call when the program starts to populate the selection screen. It will retrieve names and descriptions from the database for the available characters and enemies.

Once the player has made their selections and starts the game, there will be a second call that retrieves all of the data for each character and enemy they selected.

Stretch Goal: There will be a login system for players so that they can store selections so that they can re-try previous battles. This will require sending and retrieving player data to and from the database.

List of Steps/Activity Diagram




Start Game
View Tutorial
[Yes]
View Tutorail
[No]
Select Character / Enemies
Start Battle
Start Round
Select Action
[Yes]
Enough AP
[Yes]
Enough AP
[No]
[No]
[No]
End Turn
Game Over
[Yes]
Declare Winner

Class Diagram

Unit

Variables

Stats
- Base Strength
- Strength Modifiers {}
- Base Willpower
- Will Modifiers {}
- Base Dexterity
- Dexterity Modifiers {}
- Base Focus
- Focus Modifiers {}
- Base Defense
- Defense Modifiers {}
- Base Agility
- Agility Modifiers {}

Resources
- Max Health
- Current Health
- Max Action Points
- Current Action Points
- Max Essence
- Current Essence
- Tension

Other
- Actions {}
- Resistances {}
- Damage Bonuses {}
- Evasion Chance
- EvasionModifiers {}
- Block Chance
- BlockModifiers {}
- Status

Methods

- PerformAction(Action, Targets)
  Run **This.SelectTargets(),** then perform one of the unit's available actions.
- TakeDamage(inDamage, Caster)
  Reduces damage from attack based on defense, resistances, evasion chance, and block chance. Then reduces **CurrentHealth** by the remaining number. <Caster value only used by Enemies>
- EssenceBurn(difference)
  Reduces **CurrentHealth** based on difference
- CheckHealth()
  If **CurrentHealth** <= 0, set **Status** to Incapacitated and **CurrentHealth** to 0
  If **CurrentHealth** > 0 AND **Status** == Incapacitated, set status to active
- CheckModifiers()
  Call ChangeDuration, and removes any with a duration <= 0.
- CheckAP()
  If ap is lower than lowest action ap cost, end turn.

Action

Variables

AP Cost
Essence Cost
Caster
Targets {}
Category
Revive
Targets
PotentialTargets {}
TargetConditions

Methods

- SpendResources()
  Spends AP and increases Essence of the caster. Then if CurrentEssence exceeds MaxEssence, set current equal to max and run **Caster.EssenceBurn(difference)**
- FindTargets()
  Populate **PotentialTargets** with targets based on **TargetConditions**
- PopulateTargets()
  Run **Caster.SelectTargets** to populate **this.targets**

Character <Extends Unit>

Variables

Evade Pity
Block Pity

Methods

SelectTargets(Action)
Take Player Input to select targets from **Action.PotentialTargets** for the action being performed

Attack <Extends Action>

Variables

Accuracy
Critical Chance
Damage Ratio
Number of Hits
Attribute

Methods

- Attack(Number of Hits, Targets)
  For each target in Targets, Run **Damage()** for each **Number of Hits**
- Damage(Hit(), Crit(), MyStr/MyWill, MyBonus {}, Damage Ratio,)
  If **Hit()** returns true, calculates the amount of damage dealt to the target.
  Also calls the target's **TakeDamage()** Function.
- Hit (Accuracy, MyDex/MyFocus, MyTension, TargetAgility)
  Returns true or false based on wether or not the attack hit.
- Crit(CriticalChance)
  Returns true if the attack scored a critical hit.

Modifier

Variables
Name
Modifing Stat
Effect
Duration

Methods
- ChangeDuration()
  Subtracts 1 from each modifier duration, and removes any with a duration <= 0.

Round Counter

Variables
TurnOrder []

Methods
- PopulateTurnOrder()
  Populate TurnOrder list with all units by agility, randomizing any equal scores.
- TakeTurn()
  Iterate though TurnOrder, giving unit action.

Enemy <Extends Unit>

Variables

Aggro Table

Methods

- TakeDamage(inDamage, Caster)
  In addition to the normal effects of TakeDamage, increase the Caster's aggro in Aggro Table
- PopulateAggro()
  At the start of battle, add all chracters to the table at 0.
- UpdateAggro()
  Find the lowest value in the table, then subtract all values in the table by that number, then set the aggro of any incapacitated characters to 0.

SelectTargets(Action)
For each target in **Action.PotentialTargets**: Choose targets starting with the character who has the most aggro for the action being performed
If multiple characters have the same aggro, or if enemies are being targeted, then randomly select the target.

Support <Extends Action>

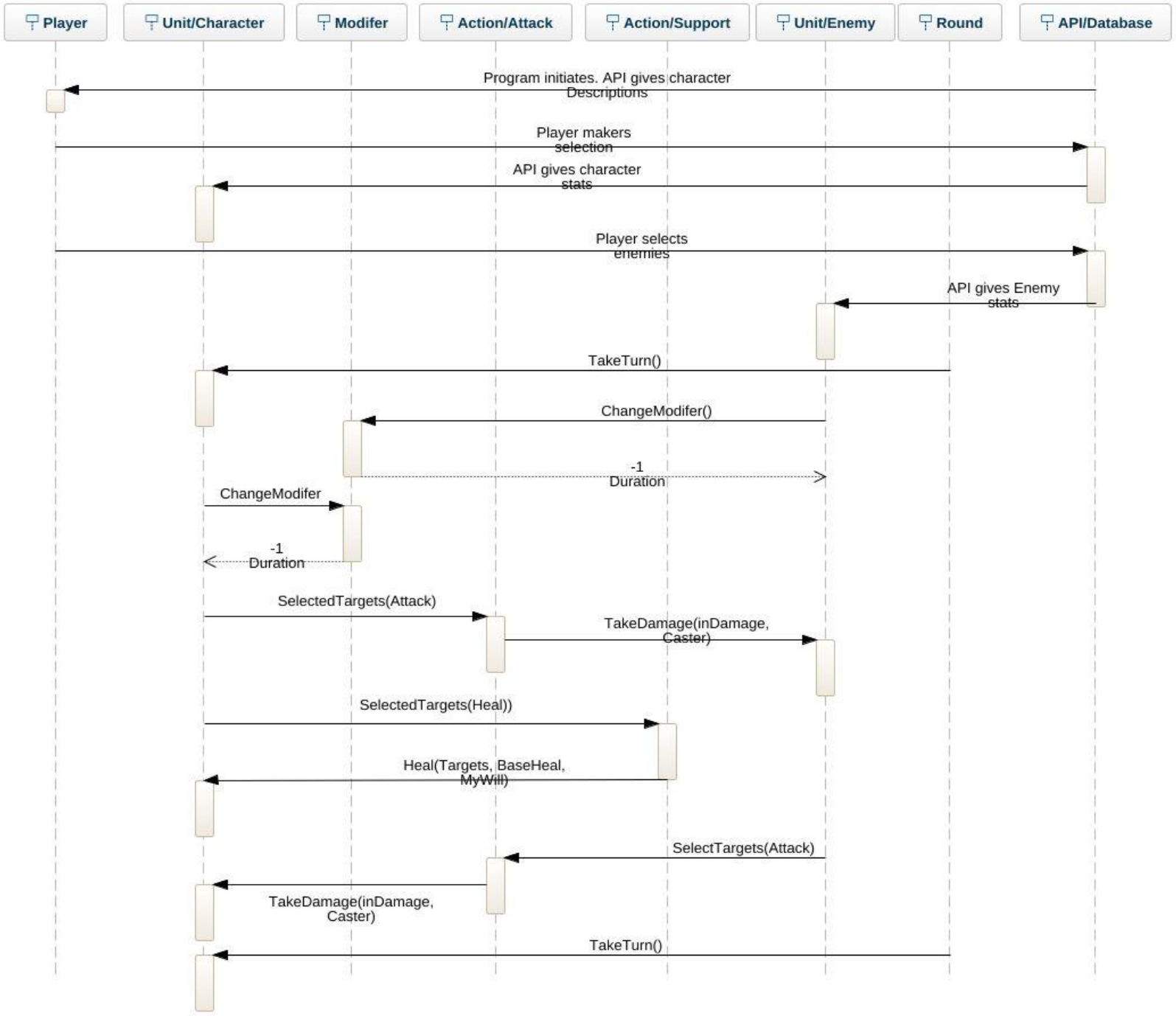
Variables

BaseHeal
ModList {
  name:
  stat:
  value:
  duration:
}

Methods

- Heal(Targets, BaseHeal, MyWill,)
  If **BaseHeal** is Not Null: For each target in **Targets** that is NOT incapacitated, Increases the target's **CurrentHealth** based on **BaseHeal, MyWill**, and a precentage of the difference between the targets **MaxHealth** and **CurrentHealth.**
- Modify(Targets, ModList)
  If **ModList** is Not Null: For each target in **Targets**, apply each modifier in ModList
- Revive(Targets)
  For each target in **Targets** that is Incapacitated: Increase the target's CurrentHealth based on **BaseHeal** and **MyWill,** and run **Target.CheckHealth()**

Sequence Diagram



Class Responsibility Collaboration chart

Unit Character	
Responsibility	**Collaborators**
Start turn	Round
Check modifiers and lower duration	Modifier
Attack enemy	Action/ Attack
Heal party character	Action/ Support
Take damage to character	Action/ Attack
End turn	Round

Modifier	
Responsibility	**Collaborators**
Check modifiers and lower duration	Unit Character
Check modifiers and lower duration	Unit Enemy

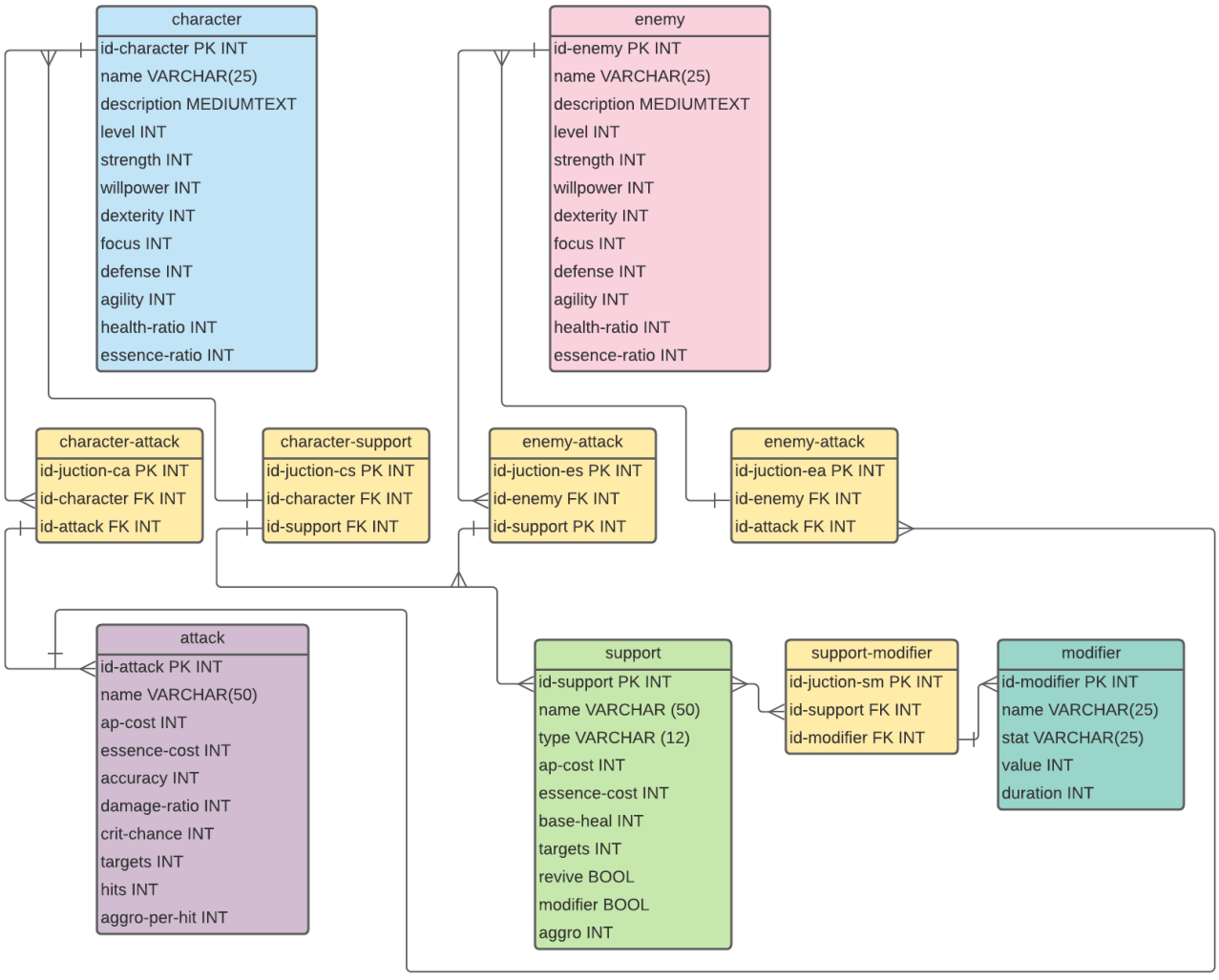
Action/ Attack	
Responsibility	**Collaborators**
Attack Enemy	Unit Character
Attack Character	Unit Enemy

Action/ Support	
Responsibility	**Collaborators**
Heal party character	Unit Character

Unit Enemy	
Responsibility	**Collaborators**
Start turn	Round
Check modifiers and lower duration	Modifier
Attack Character	Action/ Attack

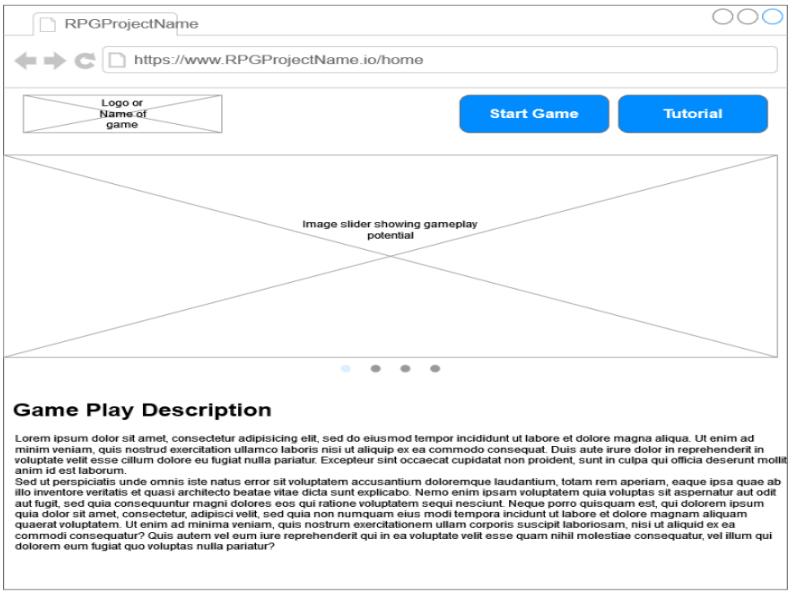
Round	
Responsibility	**Collaborators**
Start turn	Unit Character
Start turn	Unit Enemy

Entity Relationship Diagram

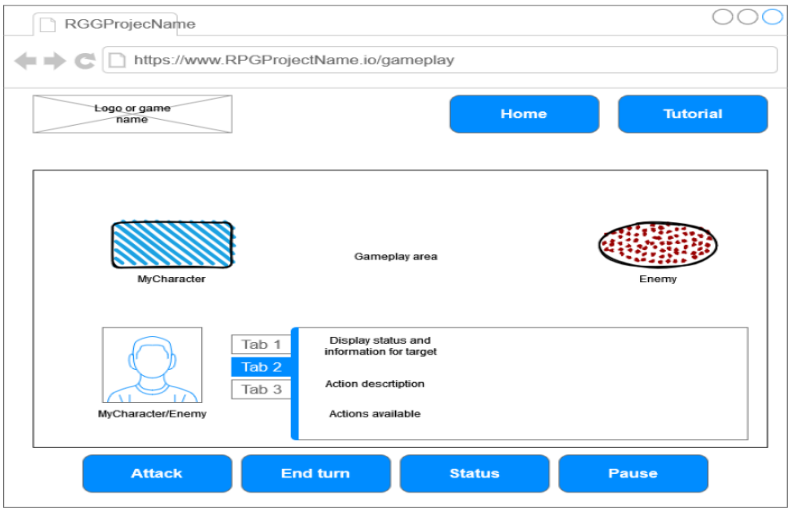


Wireframe Diagram

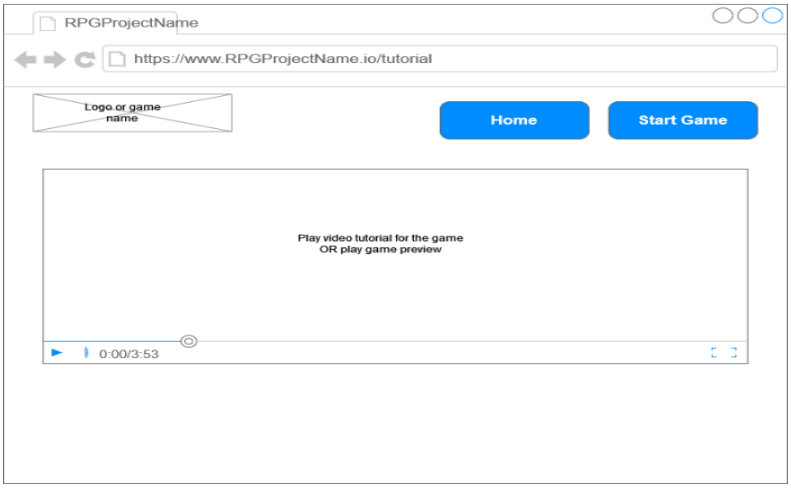
Homepage



Gameplay Page



Tutorial Page



Task Object Responsibility Chart

Object	Responsibility
Unit - PerformAction()	Call functions from the selected **Action** class to perform said action
Unit - EssenceBurn()	Take a value passed from **Action.SpendResources()** and use it to calculate how much to reduce the unit's **currentHealth** by.
Unit - CheckHealth()	Any time the value of **currentHealth** changes, set **status** to the correct value: - *good*: currentHealth/maxHealth > 0.5 - *low*: 0.5 >= currentHealth/maxHealth > 0.25 - *critical*: 0.25 >= currentHealth/maxHealth > 0 - *incapacitated*: currentHealth = 0
Unit - CheckModifiers()	Iterate through the **unit's** list of **modifiers**, call that modifier's **ChangeDuration** function, and then remove any **modifiers** that have a **duration** of 0 or less.
Unit - CheckAP()	Get the AP cost of each **action** in the unit's list of actions, then force the unit to end it's turn if it does not have enough AP to use any of those abilities.
Character - SelectTargets()	Using the input action's **number of targets**, take user input to select targets for the action.
Character - TakeDamage()	Take an integer value passed from **Attack.Damage()** and apply values from the object calling **TakeDamage()** to modify the original amount. Then reduce the object's **currentHealth** value.
Enemy - PopulateAggro()	Add each **Character** participating in the battle and add them to the enemy's **aggro table**
Enemy - UpdateAggro()	At the end of the **enemy's** turn, find the **character** in the enemy's **aggro table** that has the lowest amount of aggro. Then, subtract that amount from **all entries** in the table so that the **lowest value is set to 0** and thus becomes the new baseline value
Enemy - SelectTargets()	For each **target** from the input **action**, select the highest value in the enemy's **aggro table** that has not already been selected as a target. Randomize tied values.
Enemy - TakeDamage()	Take an integer value passed from **Attack.Damage()** and apply values from the object calling **TakeDamage()** to modify the original amount. Then reduce the object's **currentHealth** value. Then, use the attack's **aggro per hit** value to update the caster's entry in the aggro table.
Action - SpendResources()	Spend AP and Essence from the unit running the current action, and then run essence burn if the essence cost of the action exceeds their maximum essence
Action - FindTargets()	Create a list of potential targets for the action
Action - PopulateTargets()	Use the unit's select targets function to create a list of selected targets
Attack - PerformAttack()	For each target in the list of selected targets, run the deal damage function a number of times equal to the attack's specified number of hits.

Attack - DealDamage()	Run the attack's hit and crit functions to determine the amount of damage dealt by the attack, and then pass that value on to the target's take damage function.
Attack - Hit()	Use the stats from the target and the unit performing the action to determine the hit chance for the atack, then run the helper class's two random number function to determine whether or not the attack hit the target. Returns "miss" "hit" or "bonus" depending on the result.
Attack - Crit()	Uses the helper class's single random number function to determine whether or not the attack scores a critical hit based on the attack's critical chance.
Support - Heal()	If the support action has a base heal value, then increase the target's current health based on the action's base heal, the caster's willpower, and the target's missing health (max health minus current health)
Support - Modify()	If the support actions list of modifiers is not empty, add each modifier in the list to the target's list of modifiers
Support - Revive()	If the action allows targets to be revived, allow incapacitated targets to be healed using the support actions heal function. Otherwise, do not allow them to be targeted by the action.
Modifier - ChangeDuration()	Have the modifier increase or decrease it's own duration by the specified amount.
Game - PopulateTurnOrder()	Sort every unit participating in the battle into a list based on their agility value.
Game - TakeTurn()	Iterate through the list generated by the populate turn order function. Have each unit in the list take their turn when they are called by this function.
Helper - OneRandom Number()	Randomly generate an integer value between 1 and 100, and return the result *(return it in an array to be consistent with the output of two random numbers?)*
Helper - TwoRandom Numbers()	Randomly generate two values between 1 and 100, and return an array containing both results.