

GIRL SCOUT: THE COOKIE QUEST

Project Description:

GSCQ is a roguelike 'dungeon' crawler, similar to Pokemon Mystery Dungeon. The player sees from above his controlled character, the Girl Scout, who moves around looking for the exit to the next level. The girl scout can find and fight boy scouts with a menu, which has commands like attack, open inventory, etc. The stats of the two fighters - such as health, attack, and defense - determine the damage that one does to the other. The character can use items that they find to increase stats and the rate of cookie collection that the character gains via movement. Cookies can be used in stores to buy items. The player wins by beating the final enemy, the Eagle Scout, at the end. If the player fails to keep the character's health above zero, the character dies permanently and the game starts over with no saved memory from the previous game.

Features:

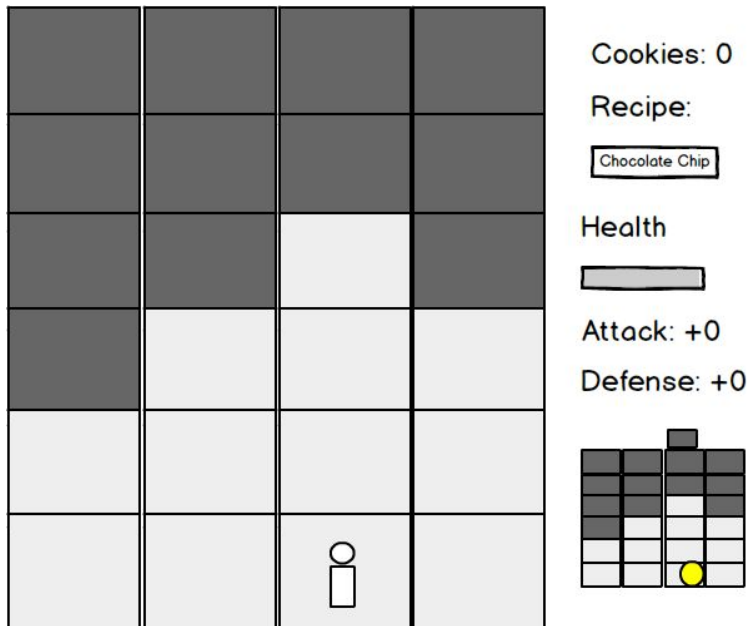
1. Randomly generates a map in the main view showing the environment that the character moves around in.
2. The main view will be augmented with a secondary view of a minimap, placed in the bottom right hand corner, which will indicate the location of the Girl Scout.
3. The Girl Scout will start the game in a random room.
4. All tiles 3 or less from the Girl Scout can be seen, meaning the items / enemies on them are visible on the main view.
5. The Girl Scout has stats like health, attack, and defense on display in a third view to the right side of the screen.
6. The default stats are for attack and defense is 0, and is 10 for health.
7. Stats can improve over the course of the game.
8. The Girl Scout can move around the map.
9. To get to the next level, the Girl Scout must find the exit, generated on the map.
10. Items can spawn on the map.
11. The Girl Scout can pick up items and store them in her inventory.
12. The Girl Scout can open up the menu view.
13. In the menu view, the Girl Scout can access the Inventory.
14. When the Girl Scout wants to use an item, she can choose them from the inventory.
Using items produces different effects on the Girl Scout, such as increasing stats.
15. The Girl Scout gains cookies, a form of currency, on movement.
16. The Girl Scout cookie default type is Plain.
17. The rate of cookie attainment increases as more cookie recipes, a type of item, are found.
18. The map will spawn one shop per floor.
19. The player can sell cookies for items or more cookie recipes. Shopping is accessed through the menu view while in a shop.

20. Enemy Boy Scouts can spawn and walk around the map. Enemies move once every time the Girl Scout moves.
21. Both Girl Scouts and Boy Scouts can start combat.
22. Girl Scouts can combat Boy Scouts using attacks, found in the menu. The Girl Scout's attack counts as her 'turn,' meaning after she attacks it is the Boy Scouts turn to move or attack.
23. Enemies can attack the girl scout. The Boy Scout's attack counts as his 'turn,' meaning after he attacks it is the Girl Scout's turn to move or attack.
24. Girl Scout can run away from battle by clicking any arrow key other than the one leading to the Boy Scout. The Boy Scout would then chase her. She must make it to the exit before the Boy Scout reaches her; else, they will continue battling.
25. If the Girl Scouts health reaches 0, a 'Game Over' screen will appear.
26. Damage inflicted upon each character in the battles depends on the stats of each character and which moves they use.
27. Have a text-driven plot, with the end goal being to defeat the Eagle Scout at the end of the game.
28. Have a start and end menu.

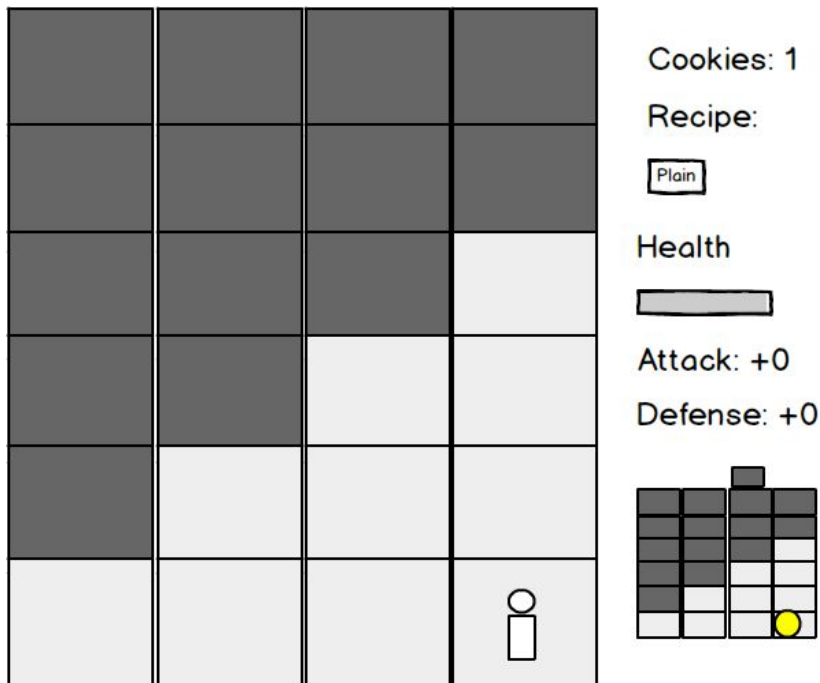
Storyboards:

Story 1: Movement

The player finds themselves in a random room of tiles.

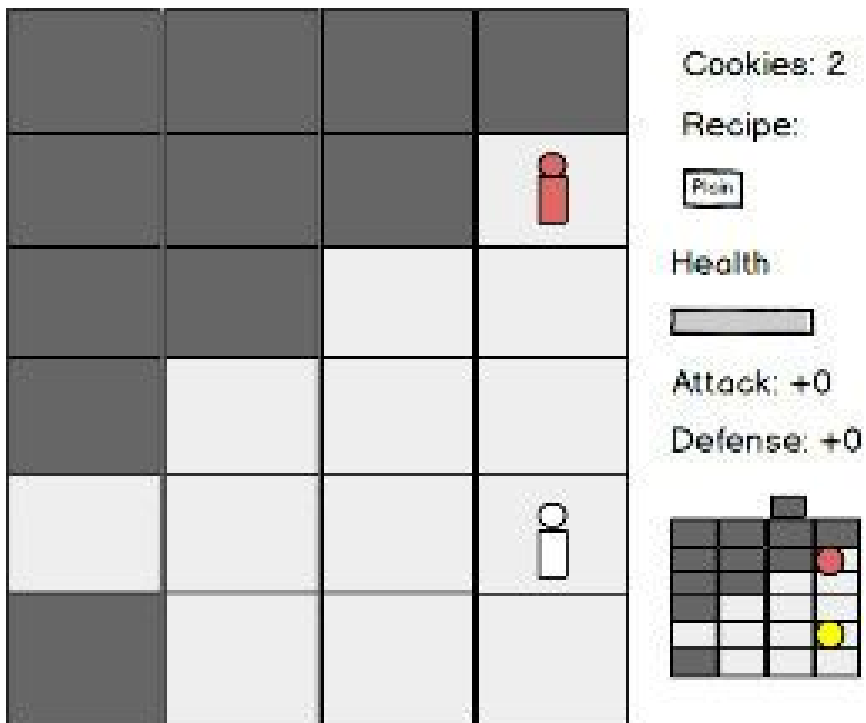


The player presses the right arrow key to move one tile to the east. The Girl Scout gains one cookie corresponding with her one movement.

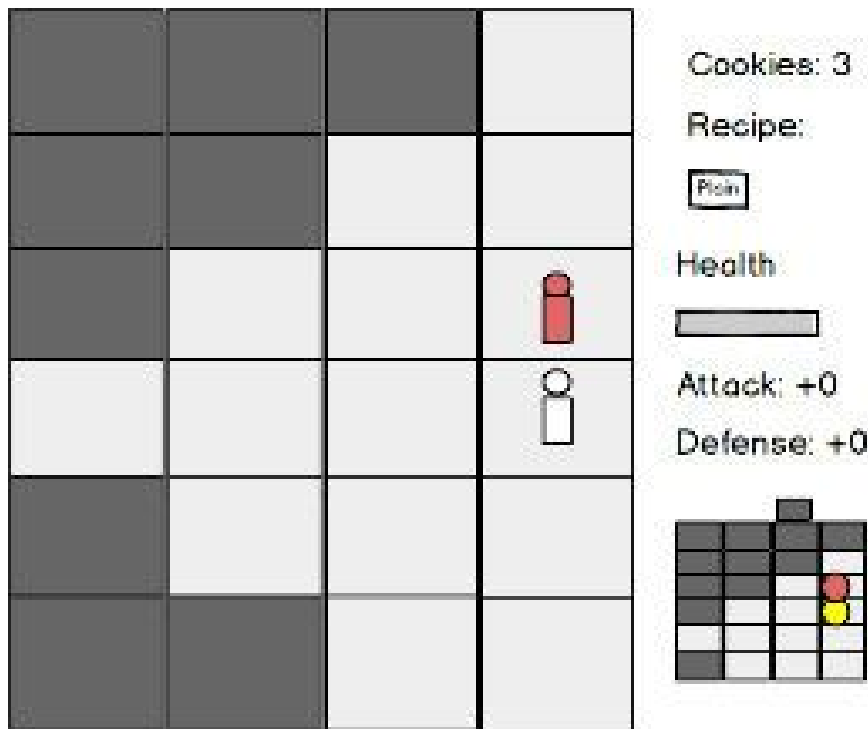


Story 2: Fighting

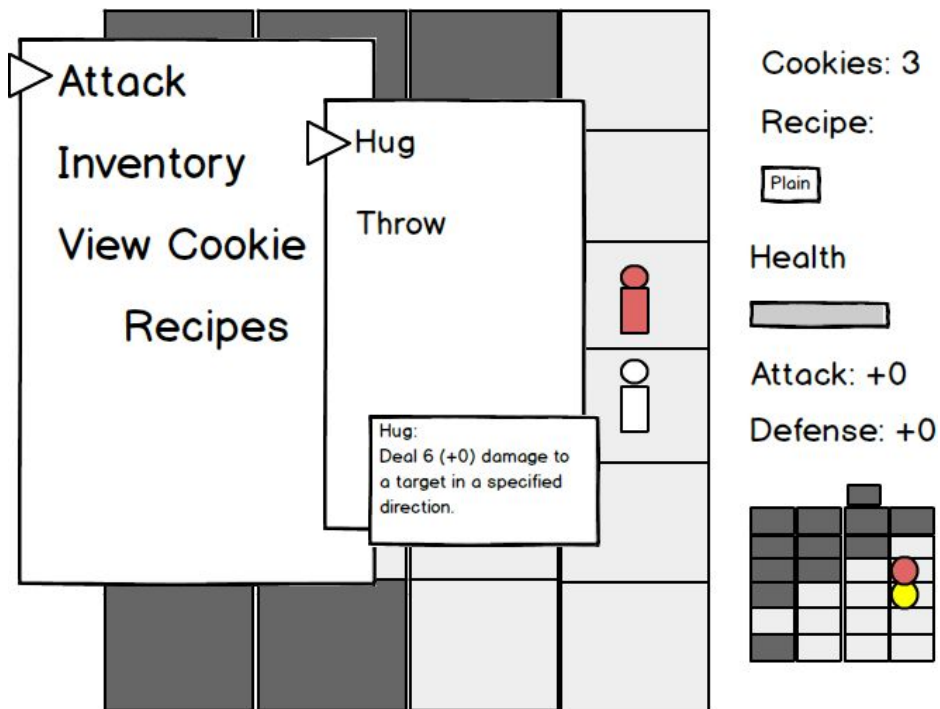
The girl scout moves north with the up arrow key, gaining another cookie, and sees an enemy boy scout two tiles above her!





The girl scout moves one tile north and the boy scout moves one tile south. They are in range for combat!



The girl scout opens up the command menu with the m key and uses the up and down arrow keys to select 'attack,' the right arrow key to move to the attack menu, uses the up and down keys to select hug, and presses enter to execute the attack.



She presses the up arrow key to indicate the direction of the attack. She hugs the boy scout for 6 damage of masculinity loss!

			 -6 HP!
			

Cookies: 3

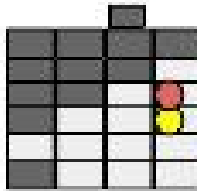
Recipe:

Plain



Health

Attack: +0

Defense: +0



The boy scout laughs derisively, dealing 3 damage to the girl scout!

			
			 -3 HP!

Cookies: 3

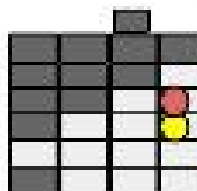
Recipe:

Plain

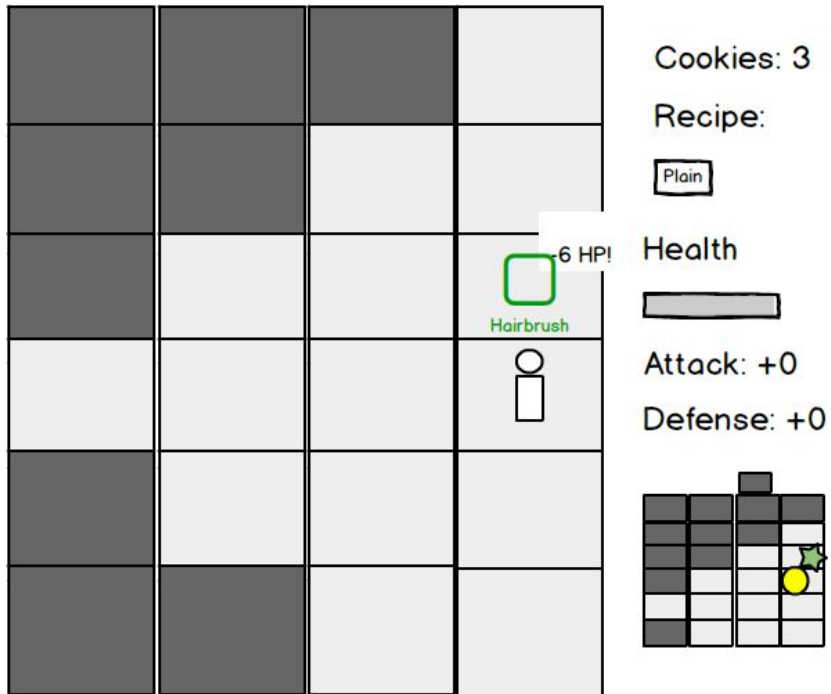
Health

Attack: +0

Defense: +0

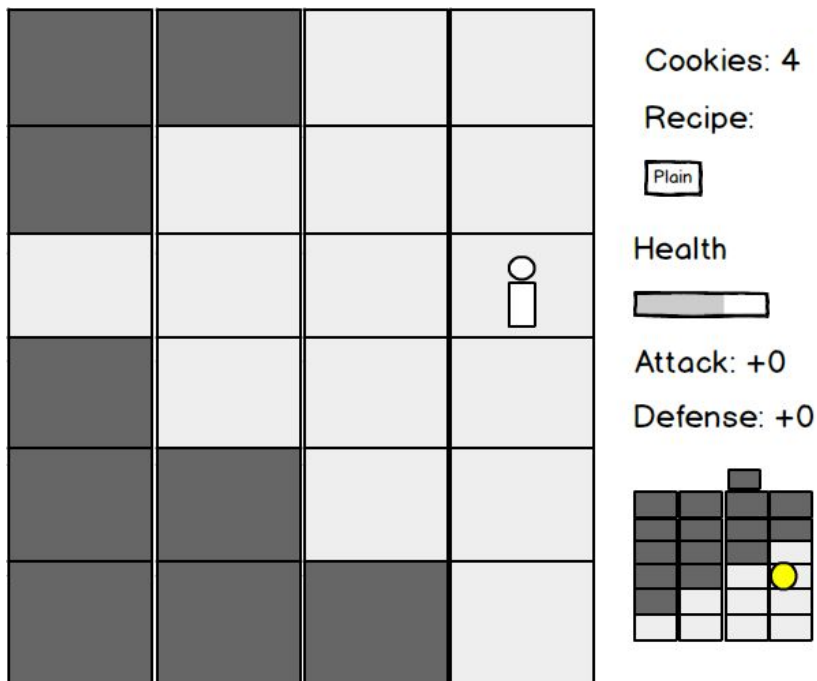


The girl scout considers walking away from this fight, her fingers hovering over the down and left arrow keys, but decides instead to once again use 'hug' for another 6 damage to defeat the boy scout. The boy scout drops the item he was carrying: the hairbrush.

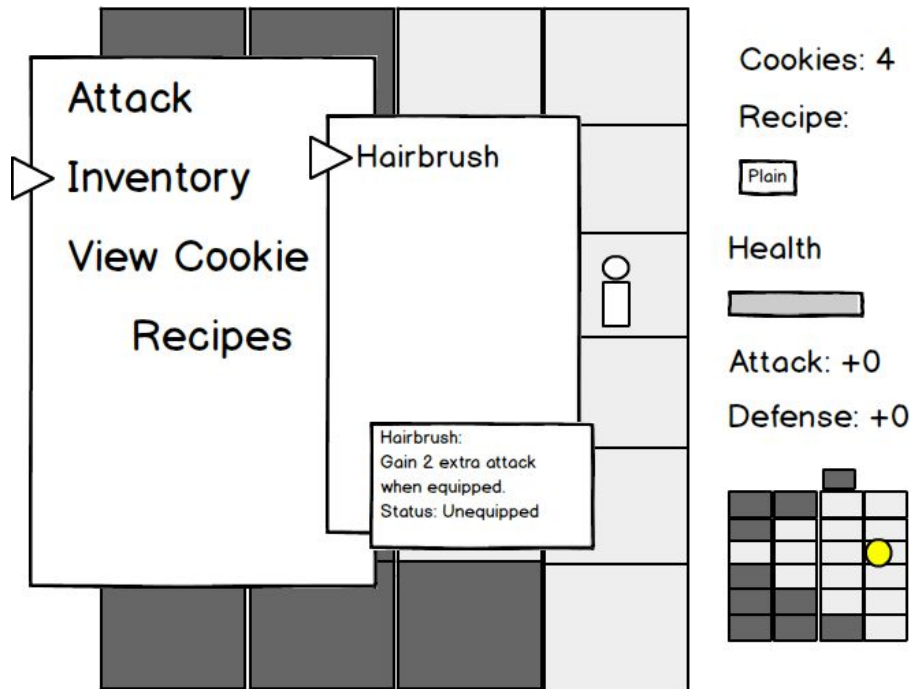


Story 3: Items

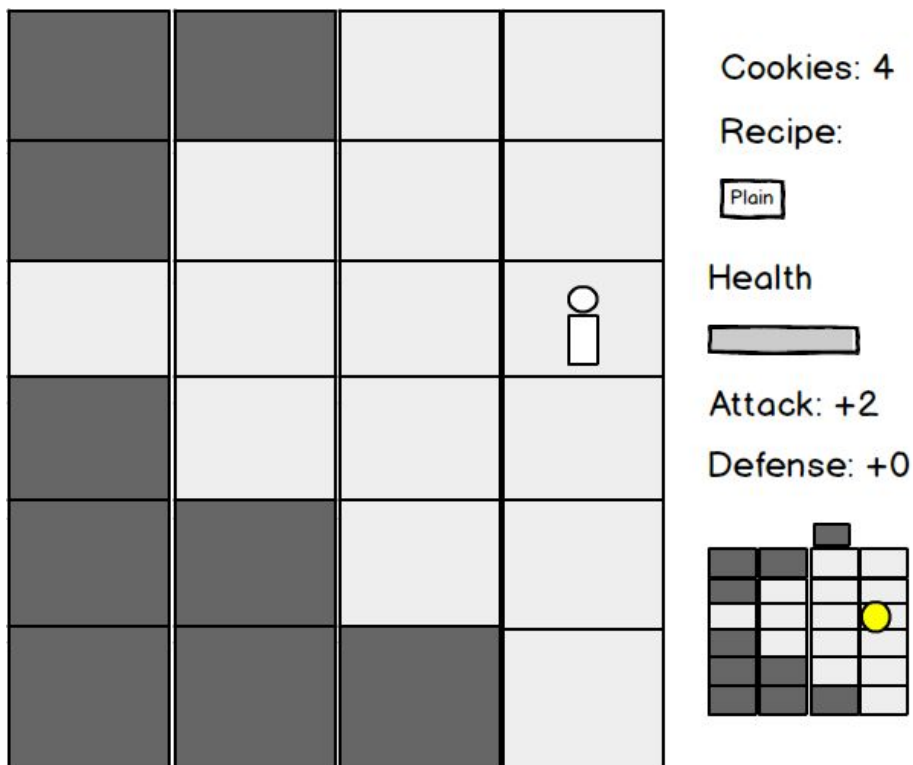
The girl scout moves with the up arrow key to the tile that contains the hairbrush, automatically picking it up.



The girl scout decides to equip the hairbrush. She clicks the m key to open up the menu, and uses the arrow keys to select 'Inventory,' then 'hairbrush.'

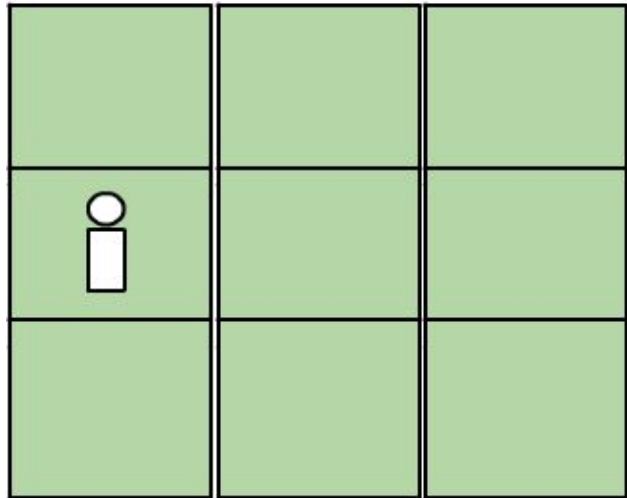


The girl scout uses the up and down arrow keys to select the hairbrush. The girl scout presses the enter key to equip the hairbrush, giving her an extra 2 attack!



Story 4: Cookies and Shops

Much later in the game, the girl scout enters a shop.



Cookies: 863

Recipe:

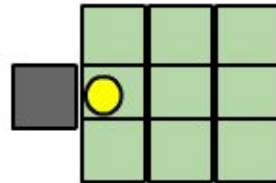
Plain

Health

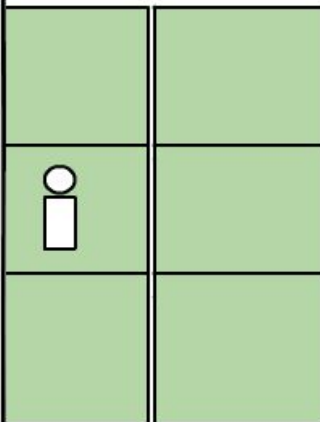


Attack: +5

Defense: +3



The girl scout clicks the 'm' key to open up the menu, and sees that a new option has been added: The 'shop' option.



Cookies: 863

Recipe:

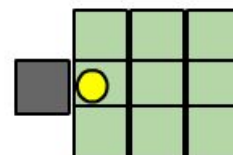
Plain

Health

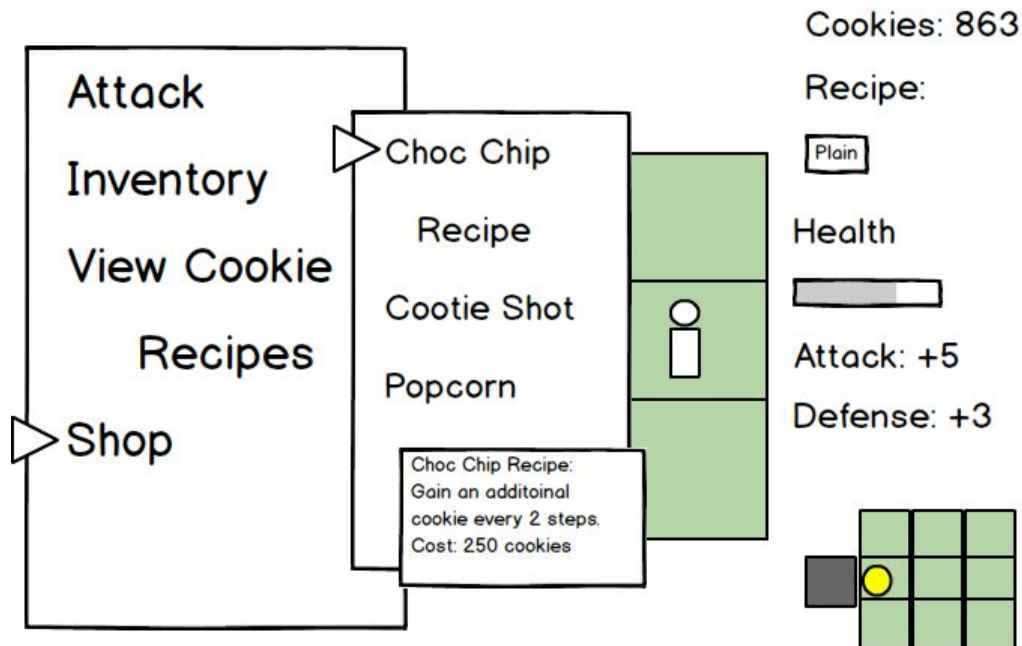


Attack: +5

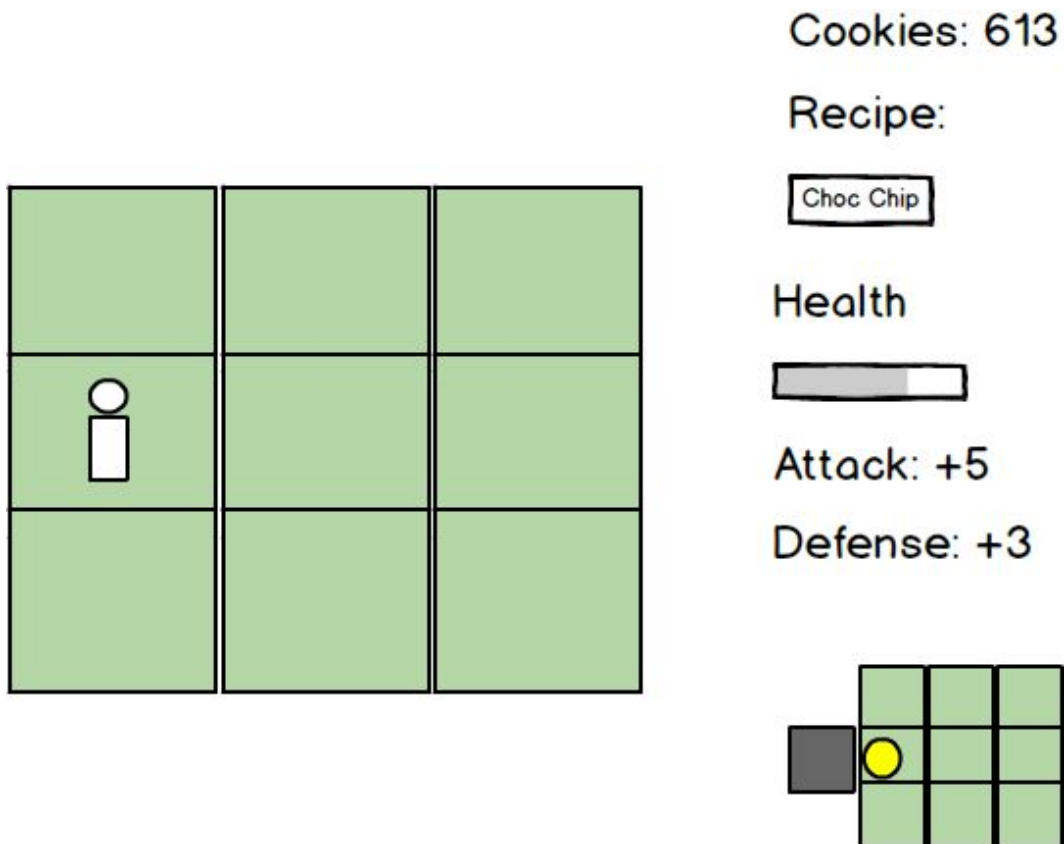
Defense: +3



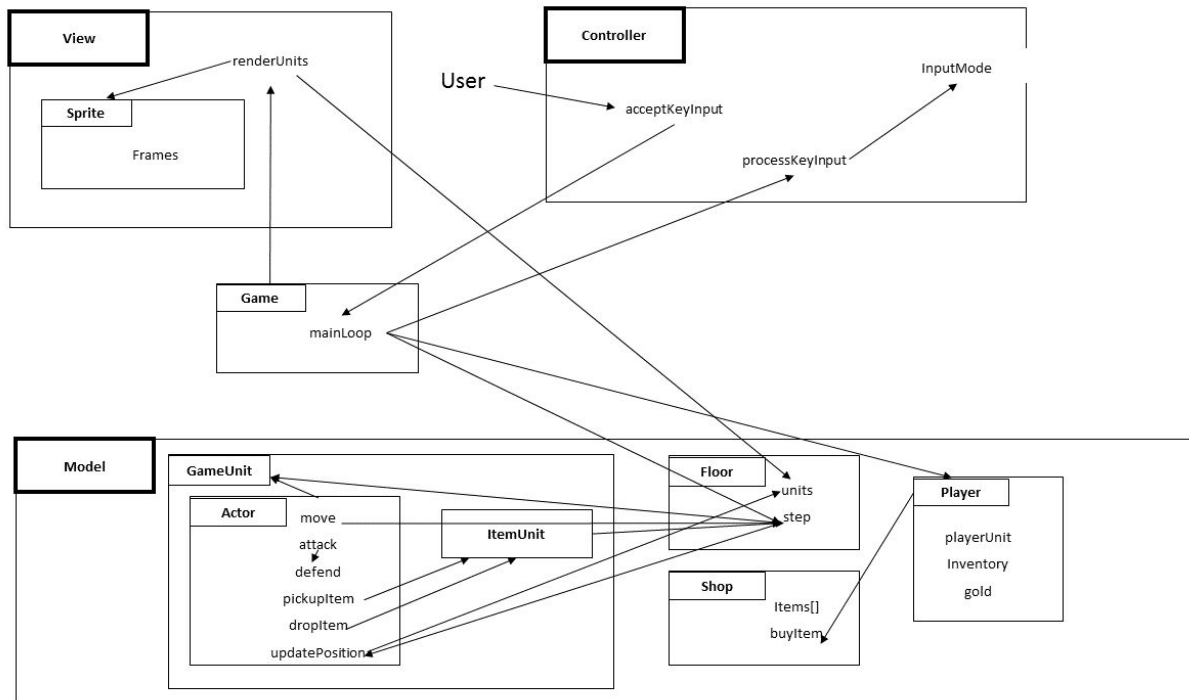
The girl scout uses the arrow keys to go to the 'shop' category.



The girl scout selects 'buy chocolate chip cookie recipe' for the cost of 250 cookies. The girl scout will now gain an additional cookie every 2 steps!



Software architecture:



Signatures:

Game - Main “wrapper” class for the underlying game. Central “hub” that controls everything.

Enum GameState - START, DEATH, END, PLAY_GAME

- Floor curr_floor - Contains the grid and unit array for the current floor. Recreated with Floor.generateRandom() every time the stairs are reached.
- Player player - See below.
- mainLoop() - Updates sprite animations (etc) and deals with input.
- changeFloor() - Refreshes the Floor object.
- GameState status - the status of the game

Model

Player - A “wrapper” class containing the player’s inventory and cookies. Also contains a pointer to the Actor that the player controls.

- ArrayList<Item> inventory - The items currently held by the player character.
- Actor playerUnit - The Actor that represents the player character on the main map.
- int cookies- the amount of cookies the player currently has.
- buyItem(Item i) - adds the Item to the player’s inventory, then subtracts the relevant amount of cookies

GameUnit - An abstract class to represent objects on the game map.

- Sprite sprite - Hook for the view to display the proper graphic.
- Point position - the coordinates on the game map of this object
- Floor owner - the Floor object this unit resides on
- getLocation() - a getter for the above

Actor extends GameUnit - Contains the data and logic necessary for units such as the player to interact with enemies.

Enum: "Direction" -> LEFT, RIGHT, UP, DOWN

- int hp, maxHP - The hit points of the actor. When they hit zero, the Actor is removed from the grid.
- int str, def - stats that affect combat
- Item toDrop - an item left on the map if this unit is removed.
- updateStatus() - Refresh this unit's position and stats to reflect changes in the Floor.
- move(Direction) - Change this unit's position.
- checkForActionsOnLocation() - See if anything can be done this tile.
- pickUpItem(ItemUnit) - put the item contained in the ItemUnit into the player's inventory
- attack(Actor, atkType) - attack the other Actor, depleting their hp
- defend(atkType) - receive an attack from another Actor. Depletes HP.

ItemUnit extends GameUnit - A "box" class to represent items dropped on the floor.

- Item item - the boxed item this unit represents/
- getItem() - getter for above

Floor - contains the grid and unit list for the current floor

- static Floor generateRandom() - Generates an entire floor (terrain map, starting units, shop/stair location).
- GameUnit[][] units - The list and locations of all units on the map.
- Tile[][] terrain - The terrain types for each square on the map
- Shop shop - The location of the shop on this floor
- int width, height - The size of the map
- Point stairs - The location of the "stairs" (the entryway to the next floor)
- generateItemUnit(Item, x, y) - Spawn an ItemUnit containing the given Item
- canMove(Actor a, x, y) - Returns whether the actor can move to the given location
- canAttack(Actor a1, Actor a2) - Returns whether the first actor is in range to attack the second.

- placeUnit(GameUnit, x, y) - Places a GameUnit at the given point
- moveUnit(x1, y1, x2, y2) - Moves a unit from one point to another
- moveUnit(GameUnit, x, y) - See above.
- runBattle(Actor a1, Actor a2) - Update the status of a1 and a2 to reflect the result of a1 attacking a2.
- step() - Runs a single “step” of the game. Each Actor will determine how to move in a non-deterministic order, then any relevant battles will take place due to each action.

Tile - data wrapper for the properties of an individual map tile.

- Image graphic - Hook for the view to display the proper graphic.
- boolean inView - If the tile is in range to be seen by the player.

Shop - Data wrapper for the shops as described in the storyboard

- Item[] inventory - The items contained in this shop.
- Point topLeft, bottomRight - The boundary points of what's considered a “shop”. Shops are guaranteed to be rectangular.

Item - Wrapper containing

- int price -
- Function effect - Returns a first-order function object containing the effect of holding or using this object.

Controller

Enum “InputMode” -> MENU, MAP

- acceptKeyInput() - If a key is pressed on the keyboard, add it to the processing queue
- Function processKeyInput() - Take a key from the processing queue and return a function detailing the result based on the game state and the value of “InputMode”

View

- renderMap() - draw the map tiles on the current screen
- renderGameUnits() - draw the sprites on the current screen

Sprite - contains a graphic that requires animation

- Image[] frames - the frames for the animation
- Point position - the position of the sprite on the screen

Tests:

Floor generation:

allRoomsMustBeConnected()

floorMustHaveAccessibleShop()

floorMustHaveAccessibleExit()

floorProperlyGeneratesItems()
floorProperlyGeneratesEnemies()
actorCanNotBeInWall() //players and enemies can't move through walls
itemCanNotBeInWall() //item can not generate in wall
boyScoutCanNotUseExit() //floor doesn't regenerate when a boy scout stands on the exit
boyScoutDoesNotDisappearUponStandingOnExit() //There is no escape...
onlyAppropriateTilesAreInView()
floorRegeneratesUponGirlScoutReachingExit()

Game Status:

gameStatusIsCorrectWhenGameLaunches()
gameStatusIsCorrectWhenGameStarts()
gameStatusIsCorrectWhenGirlScoutDies()
gameStatusIsCorrectWhenGirlSchoutWins()

Stats:

girlScoutNeedsDefaultStatsSetCorrectly() // default attack and defense are 0
girlScoutNeedsDefaultHealthToBeTen()
boyScoutsNeedsDefaultStatsSetCorrectly()
itemProperlyAugmentsStats()

Items:

girlScoutAutomaticallyPicksUpItem() //test to see if an item is removed from the map when the Girl Scout stands over it
girlScoutAutomaticallyStoresItem() //test Inventory size when an item is picked up
girlScoutCanEquipItems()
girlScoutCanUseItems() // test stats when items are used
itemIsRemovedFromInventoryWhenUsed() // test Inventory size when item is used

Shop:

itemAddedToInventoryWhenBought // test Inventory size when item is bought
cookiesSubtractedWhenUsedToBuyItems // test cookie counter when item is bought
shopGeneratesWithItems() //tests size of Items in Shop ArrayList

Combat:

boyScoutTakesDamageBasedOnStats() // test that appropriate damage is dealt based on Girl Scout's attack and the Boy Scouts defense
girlScoutTakesDamageBasedOnStats() //test that appropriate damage is dealt based on Boy Scout's attack and the Girl Scout's defense

boyScoutAttacksGirlScoutWhenInRange()

boyScoutDoesNotAttackOutOfRangeGirlScout //tests if the Boy Scout moves instead of attacking if the Girl Scout is out of range

actorRemovedWhenHealthEqualsZero()

Cookies:

cookiesGoUpByOnePerMovementWithOnlyDefaultRecipe()

cookiesGoUpBy--Per--MovementWith--Recipe() //Test for all recipes

cookiesGoUpBy--Per--MovementWith--And--Recipes() //Test to ensure two recipes stack

cookiesGoUpBy--Per--MovementWith--And--And--Recipes() //Test to ensure recipes stack with more than two recipes