Michael Knutson **Coeur d'Game**

1. Brief introduction __/3

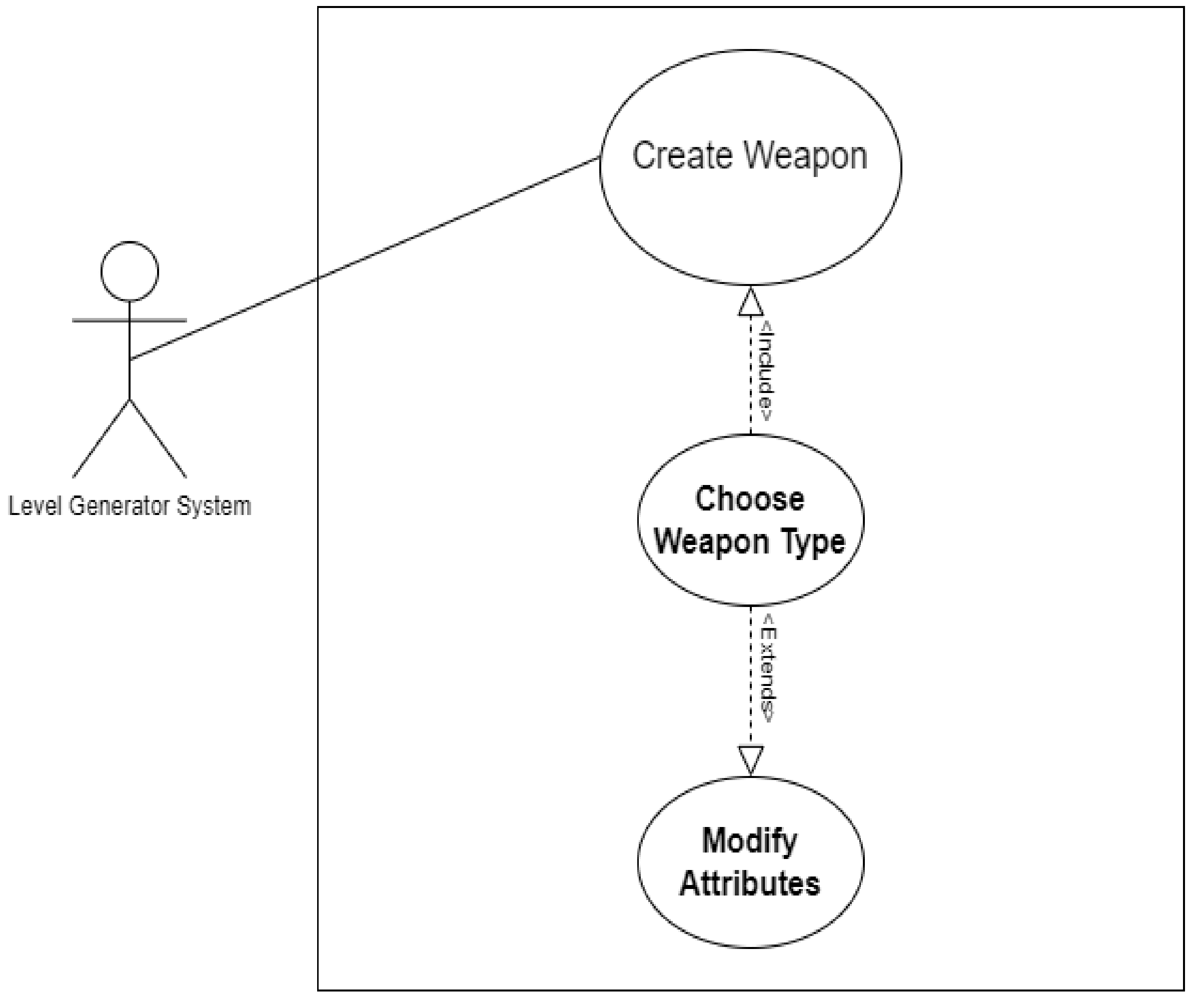
My feature for our video game, Rogue Realm, is to create items that the player will pick up and interact with while playing our game.

To ensure that our level generator system and our video game play is successful, I will be creating many different items that the user will use and interact with during the game. The items I will be creating will be in three different categories, pedestal items, chest items, and crate items. Pedestal items will be different types of weapons such as range attack weapons, melee attack weapons, and magic weapons. Chest items will be categorized as power-up items such as maximum health upgrades for the player, speed boost for the player, jumping power for the player, and an increased damage boost for the player, just to mention a few. Crate items will be a mixture of health boosts that will increase the health of the player, and coins that will give the player points that will be used when the player dies and will be factored into the players final total score.

I am responsible for creating many different instances of these items and coming up with the attributes for these items which will directly interact and affect the player using these items. For the weapons items, there will be 3 different categories, range attack, melee attack, and magic attack, that will each have their own abilities and stats. When a player uses these weapons, I will need to encapsulate the data needed for the player system to be able to use those weapons during gameplay. For the power-ups, I will need to encapsulate the correct data that the player system will need in order to use those power-ups during gameplay. For the health and coin items, I will also need to make sure I encapsulate the correct data for the player system to use in order to correctly impose those benefits to the player during gameplay.

2. Use case diagram with scenario __14

Use Case Diagram



Scenarios

Use Case #1:

Name: Create a weapon for the level generator system to use within the map.

Summary: The level generator system requests a random weapon to place in the game at a certain location within the level when a weapon item is chosen from the system. The create a weapon use case will then randomly choose which type of weapon to return back to the level generator system along with the correct attributes for that specific weapon type.

Actors: Level generator system.

Preconditions: The level the player is playing is passed through the weapon creator use case as it affects the attributes of the weapon that is returned. A list of weapon types and weapons of those types are already created and in a database waiting to be requested by the level generator system.

Basic sequence:

Step 1: The level generator system will request an item to be created and put in a location on the map.

Step 2: The weapon creator system will randomly select which type of weapon to use.

Step 3: The weapon creator system will then use the randomly selected weapon type to choose a specific weapon from the selected weapon type list.

Step 4: The specifically selected weapon will then use the player level value (what level the player is playing on) to determine if the weapon needs to increase the stat values for that weapon.

Step 5: Once the weapon is selected and the attributes are modified (if need be), the weapon creator system sends the completed weapon to the level creator system.

Exceptions:

Step 1: The player already has a weapon and the level generator system is requesting a new weapon for the player (because the player chose to swap weapons).

Step 2: The player will swap weapons with the newly randomized weapon they are trying to pick up and the weapons' attributes will be applied to the player via the game manager system.

Post conditions: The player will be able to use the weapon they pick with the updated attributes of that weapon.

Priority: 1*

ID: CW1

*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

3. Data Flow diagram(s) from Level 0 to process description for your feature ______14

Data Flow Diagrams

Diagram 0:

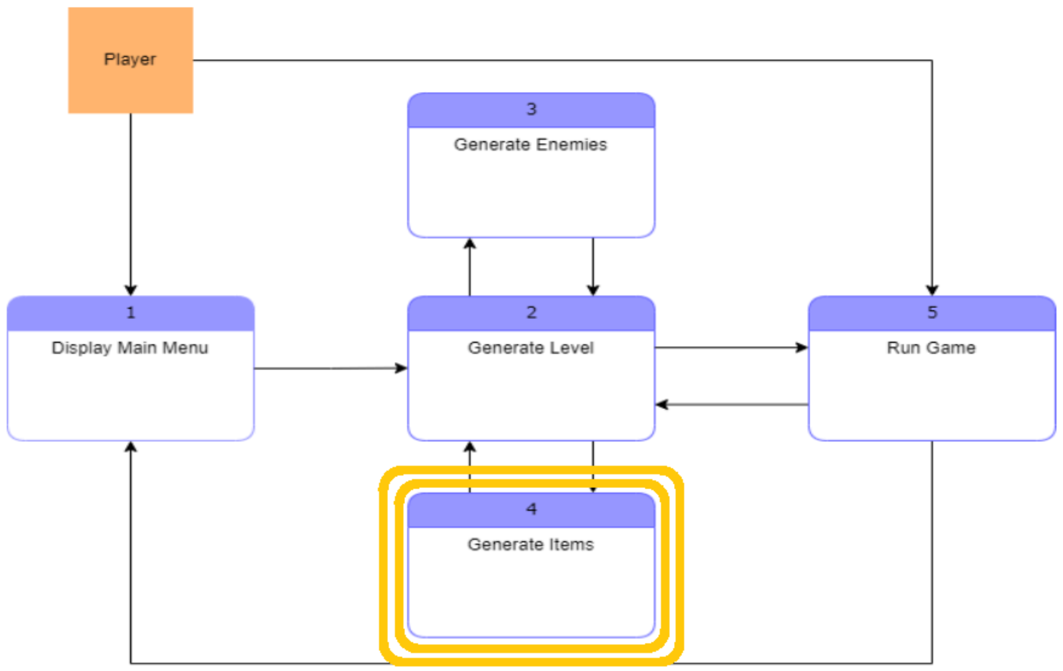
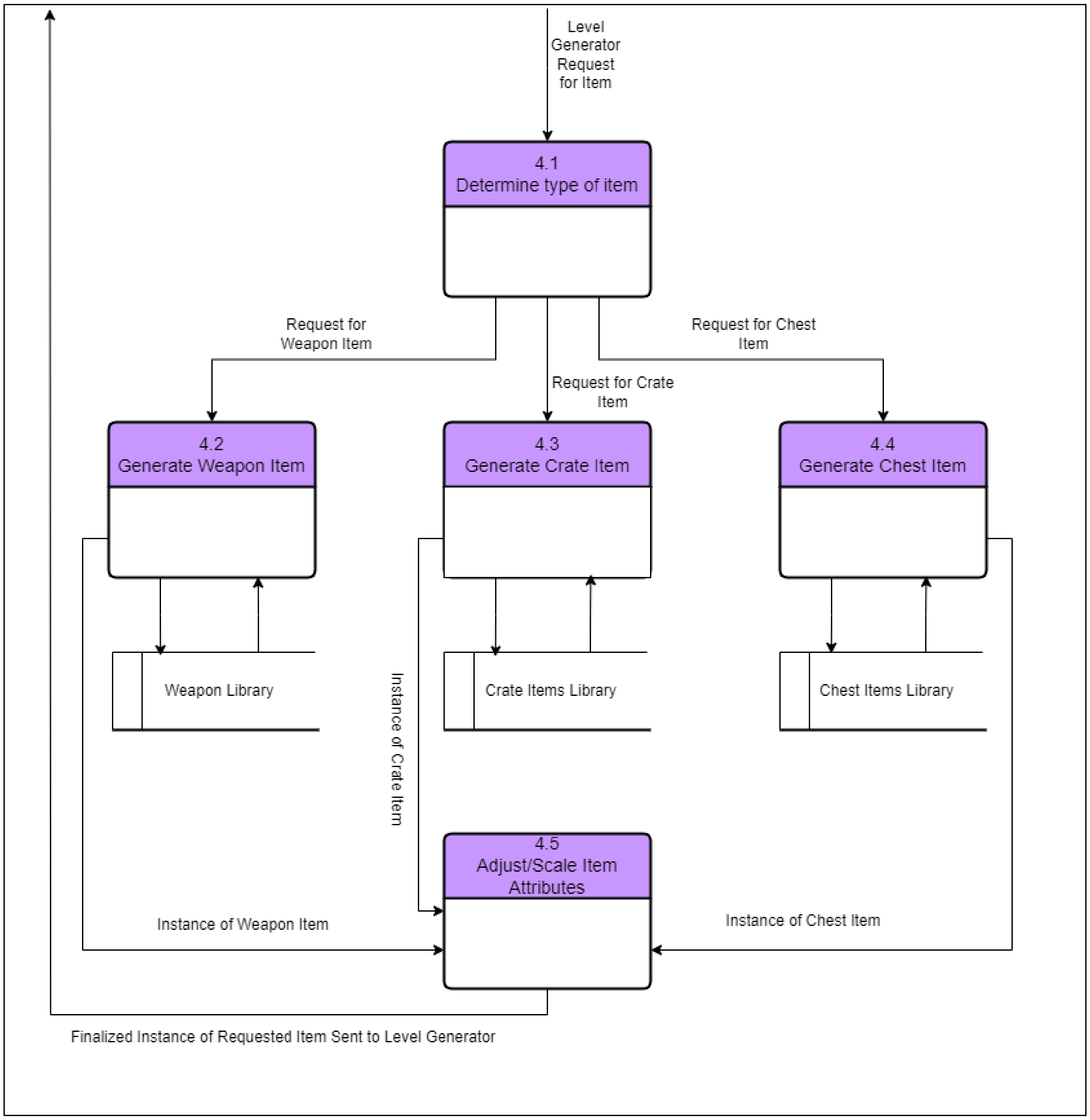
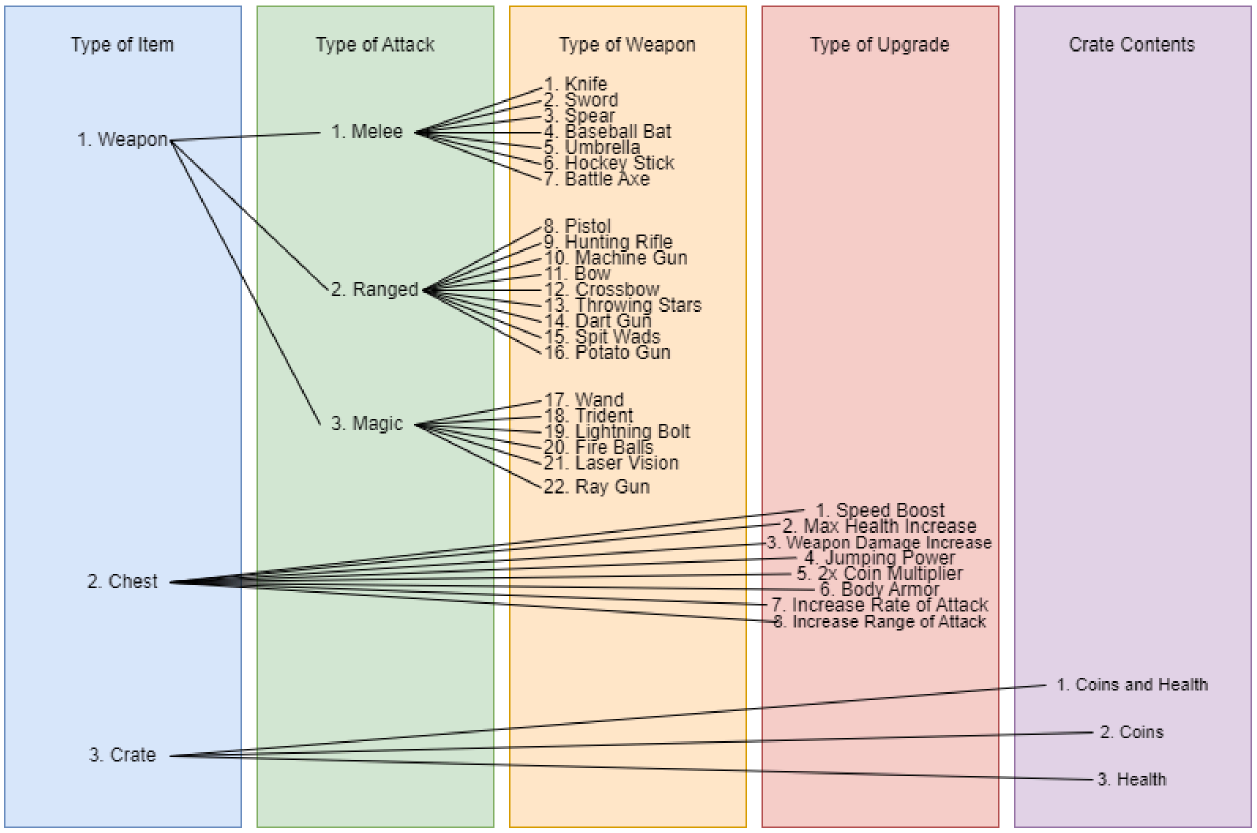


Diagram for Generate Items(4):


Level Generator Request for Item
4.1 Determine type of item
Request for Weapon Item
Request for Crate Item
Request for Chest Item
4.2 Generate Weapon Item
4.3 Generate Crate Item
4.4 Generate Chest Item
Weapon Library
Crate Items Library
Chest Items Library
Instance of Crate Item
4.5 Adjust/Scale Item Attributes
Instance of Weapon Item
Instance of Chest Item
Finalized Instance of Requested Item Sent to Level Generator

Process Descriptions



4. Acceptance Tests ______9

Example for random item generator.

Generate 100 instances of each item (a weapon item, a chest item, and a crate item) and send output of attributes to a file.

The output file will have the following characteristics:

- Item type (weapon, chest, or crate)
- If weapon item:
  - weapon type
  - range of attack attribute
  - attack damage attribute
  - rate of attack (or fire) rate
  - player speed with weapon attribute
- if chest item:
  - speed increase attribute
  - max health increase attribute
  - weapon damage increase attribute
  - player jumping power attribute

    - coin multiplier attribute
    - body armor (additional health) attribute
    - rate of attack increase attribute
    - range of attack increase attribute
- if crate item:
    - # of coins (between 1 and 20) and # of health points (between 1 and 10)
    - # of coins (between 1 and 25)
    - # of health points (between 5 and 15)

The acceptance test will make sure that for each possible type of item to be picked up, whether a weapon item or a chest item or a crate item, there are accurate and essential stat attributes that are generated for each possible type of item. A melee attack weapon type should not exceed a value of 4 range attack attributes. A range weapon type should not have a value of less than 4 for a range attack attributs. A magic weapon type should not have a value of less than 3 for a range attack attribute. The testing scenario should make sure that for each of the weapon type attributes, range of attack, attack damage, rate of attack, and player speed with weapon, are within the ranges that are predefined for each weapon type.

The acceptance test should also test that the player speed, player jumping height, player health, and player weapon damage is within a range that is comparable with the expected range (a predefined min and max value) declared for these attributes.

The acceptance test should also test whether the number of coins and/or health points randomly generated are within a certain predefined range for the given category of crate items. If coins and health points are generated together in the same crate, the number of coins should be between 1 and 20, and the number of health points should be between 1 and 10. If coins are only generated within a crate, the acceptance test should test to see if the number of coins generated are between 1 and 25. If health points are only generated within a crate, the acceptance test should test to see if the number of health points generated are between 5 and 15.

Example for Item Generator System:

Item Type	Type of Item	Range of Attack	Attack Damage	Rate of Attack	Player Speed Increase	Jump Power Increase	Max Health Increase	Coin Multiplier	Body Armor	# of Coins	# of Health Points
Weapon	Range	5	5	3	0	0	0	0	0	0	0
Weapon	Magic	4	6	5	0	0	0	0	0	0	0
Crate	0	0	0	0	0	0	0	0	0	5	6
Chest	0	1	0	0	0	0	0	0	0	0	0
Crate	0	0	0	0	0	0	0	0	0	7	3

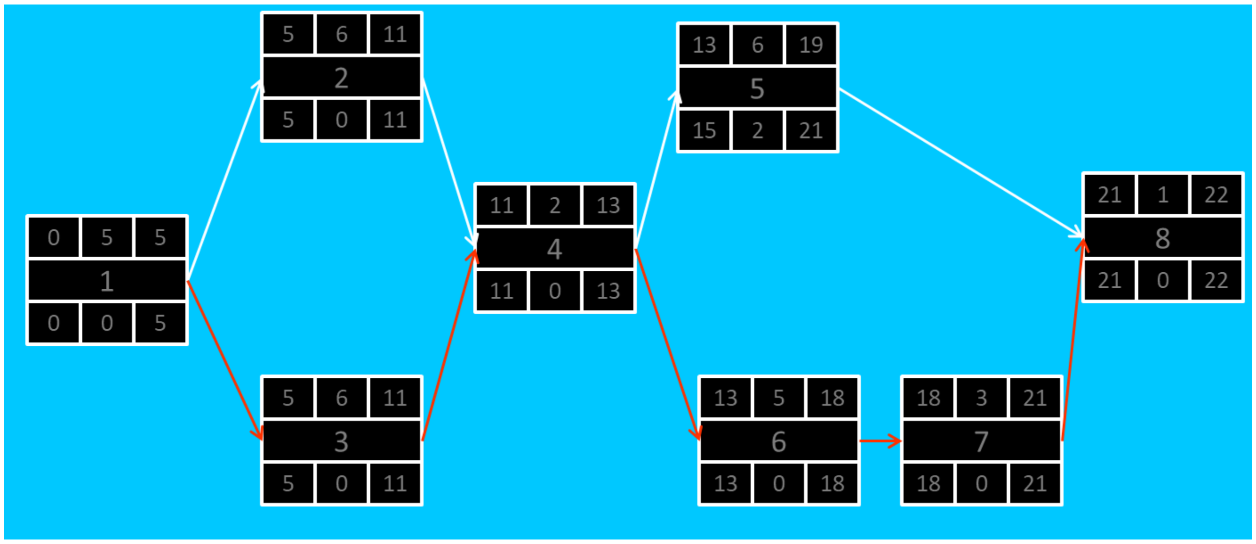
Crate	0	0	0	0	0	0	0	0	0	0	12
Weapon	Melee	1	2	6	0	0	0	0	0	0	0
Chest	0	1	0	0	0	0	0	0	0	0	0
Chest	0	0	2	0	0	0	0	0	0	0	0
Crate	0	0	0	0	0	0	0	0	0	10	0
Weapon	Melee	2	4	4	0	0	0	0	0	0	0
Chest	0	0	0	0	2	0	0	0	0	0	0
Chest	0	0	0	0	0	0	2	0	0	0	0

5. Timeline ________/10

Work items

Task	Duration (Weeks)	Predecessor Task(s)
1. Brainstorm types of items	1	-
2. Brainstorm item list	1	1
3. Brainstorm attribute list	1	2
4. Create database for items	3	3
5. Create pixel art for each item	3	4
6. Programming	3	5
7. Testing	2	4
8. Installation	3	5

Pert diagram



Gantt timeline

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	***														
2		***													
3			***												
4				***	***	***									
5							***	***	***						
6										***	***	***			
7							***	***							
8										***	***	***			