**Level Platforms:**

**Platform Management**

When playing a level, only a small amount of platforms will be kept active in order to efficiently manage resources. A list of all platforms in the level as well as other auxiliary lists are kept using data structures of minimal size. The actual visible platforms are kept in a list of instantiated transforms which shares resources with a queue of inactive transforms. As the player moves the screen along with their motion, events are run in the update function to activate and deactivate platforms as necessary.

Initial level setup runs in O(n)

Updates where no events occur run in O(1)

Events for a potential platform activation check run in O(1)

Events where a platform becomes active run in O(1)

Events where a platform becomes inactive run in O(k)

where n is the number of platforms in the level and k is the number of instantiated platforms

Total space used is O(n) platforms and O(k) transforms

**Level Creation**

Level creation can be primarily handled by a system which will procedurally generate platforms to make up a level. Once such a level has been made, it can be used as-is or it can be modified manually to allow for further touches to be added. Criteria used for the generation of the levels remains to be decided.

An important aspect of the generation of levels is ensuring that the level which is produced is feasible in terms of a player reaching the goal. In order to ensure that this is possible, the generation of a level simultaneously constructs a graph in which the nodes are the platform which have been created and the edges represent a gap between two platforms which can potentially be travelled. The edges do not necessarily indicate that it is possible to travel between two platforms, but rather that there is a good chance that it is possible. The reason for constructing this graph is to then have a rather general structure which can be queried for various operations, most importantly checking if a specific player can reach the goal. Each edges will be given various weights measuring different aspects of the difficulty associated with travelling along that edge. A player has specific attributes associated with their abilities to progress through an area. By using the abilities of a player as parameters, a search algorithm can be run on the graph to check if a specific player can reach the goal. Specific weights and player abilities remain to be decided. Other applications of the graph remain to be decided as well.

Graph construction time is O(n2)

The space used by the graph is upper bounded by O(n2) but is expected to be O(nk) in practice

**Current Systems/Concepts:**

**Characters**

The character class is the highest level of abstraction for the characters of the game which abstracts into playable characters, enemies, etc. Characters all have basic stats including their health, attack power and any other concepts which are common to all characters. Currently characters are set up as being highly stat-based, allowing for a levelling system with many different statistics. Depending on the direction is which the game is taken, this can be further developed or it can easily be cut out to lessen the dependence on stats. In this way, the game can easily be made with or without any degree of rpg-styling.

**Character Types**

The character types define how a character primarily will interact with their environment, both in terms of movement and combat. Currently, two types are proposed as being an environment manipulation type and a mobility type. To give the game a cooperative aspect, it can be designed in a way that requires the cooperation of different character types in order to progress.

**Character Classes**

Character classes are an optional sub-categorization of playable characters to allow for more variety in terms of play style. Each character class will belong to one of the character types. His allows players to form teams which satisfy the required player types while giving them different class options. Class options could potentially allow for choices between range-based classes, melee-based, magic-based, etc or they could be even more specific than this.

**Status Conditions**

Status conditions are conditions like bleeding, poison, crippling which can be applied to characters through the use of various attacks, skills or special weapons. Conditions will either damage or hamper the character in some way for a period of time.

**Attacks**

All characters have specific attacks which they can use. The attacks available to them are based on their character type and class. Characters could potentially be allowed to learn more attacks as they progress through the game. Attacks have a number of different characteristics which define their nature, including the range, speed, accuracy, etc. Attacks can also have special properties such as explosions, arcing shots, ability to inflict conditions, etc.

**Skills**

Similar to attacks, the skills which a character can use are determined by their type and class. Skills comprise the non-combat abilities of characters, particularly the abilities they will use to move through the levels. Skills could also incorporate buffs or other passive effects. A system could also be incorporated to allow characters to level up or customize their skills.

**Weapons**

Currently there is only a parent class for weapons but this can easily be sub-classed into specific weapon types. The types of weapons usable by a character could potentially be determined by their type and class. Weapons have particular attacks which they can be used for. In order to use a weapon attack, the weapon must have that type of attack and the character must also know that particular attack. Weapons can be given a number of different stats which combine with the stats of the attack and the character’s stats to determine final efficiency, power, etc of the attacks made by a player using the weapon. Weapons are also currently set up as being one-handed or two-handed to allow players to have the option of dual-wielding or equipping something else in their other hand.

**Equipment**

Currently equipment is limited to being other items or accessories which can be equipped in the character’s other hand. This can easily be expanded to allow for a character to fully equip themselves with armor or other things.

**Crafting and Salvaging**

Items are given salvage values which indicate the materials that will be obtained if the item is salvaged. Some items will also have crafting costs. The crafting costs are the materials required in order to craft that item.

**Upgrading**

A variety of weapon upgrades can be made to give to weapons. Upgrades have specific costs such as materials or money associated with them and may also have prerequisites of other upgrades which must be applied first. Each weapon indicates which upgrades can be applied to it. Upgrades can be used to modify weapons to give them special properties, change the type of attacks that the weapon can use or increase its efficiency.

**Shops**

Different types of shops allow for players to preform different actions. The merchant shop allows for items to be bought and sold. The smith allows for salvaging and crafting to be done. The workshop allows for upgrades to be applied to weapons.

**Enemies**

Enemies are subclassed from the character class and are given addition attributes which define their behavior and abilities. Enemies can be given various abilities which might allow for them to fly, swim, climb up parts of the level, etc. Their behavior includes how aggressive they are, whether they patrol or stand still, how far from their original location they will stray, etc. Enemies are also given senses in terms of vision and hearing to determine how they will detect the players.

**AIs**

Some portions of the behavior of enemies are defined through the type of AI that they have. This will primarily handle their combat tactics. Different AIs will be written to allow for a variety in this area. AIs could also potentially be applied to bots to act as teammates for a single player version of the game.