**Proposal**

**Description:**

The Game is the title of the miniworld and does not exist as a separate miniworld entity. The player’s objective is to conquer planets and gather resources in their section of the galaxy by commanding allied combatant ships from their megaship. The player loses if all their ally ships are defeated or if their megaship is destroyed.

The game has users who play the game. Each user has a unique account with a username, a password stored as a salt and hash**,** volume settings, and a set of saved games. Volume settings contain multipliers for sound controls, including a global, music, menu, and effects volume. Each saved game has identifying information such as a timestamp, a name, and an ID. The ID is unique to each game save; names can be the same for different saves. A saved game lists the state information needed to load and save a game: each saved game also has a game progress state, a save parent, and a list of map objects. The possible states for the game progress state are ‘in progress’ and ‘lost’.

The game also has new map templates that are used to instantiate a new saved game. Each new map template has a template number, a version number, and a saved game it clones to instantiate new maps.

Map objects represent everything that has a drawable position in a given game state. Each map object instance has a unique ID to separate it from other map objects, as well as a list of resources that forms an inventory. All map objects need a transform component with and a position.Stars and Planets don’t need rotations, however. A position has an x and z coordinate. A map object can be a star, a ship, a planet, or a weapon emission. All map objects also have an entity type identifier that is used by the application’s processing.

A star has a name; the application uses the star’s entity type to calculate the star’s unique attributes.

A planet has a name, a relationship state, a life state, a planet guard set, and a set of planetary resources. The planet’s relationship is enemy until all planet guards have been slain, after which it will be friendly. A planetary resource has a type, which is a weak primary key, a maximum resource count, a current resource count, and a regeneration rate. A planet guard set is a list of planet guard combatant ships. A planet has a load flag which it uses to determine if it is being loaded for the first time and must initialize its planet guard. A planet is alive until all its resource counts reach 0, at which point it dies (life state is either alive or dead).

A ship is a map object, which can be either a megaship or a combatant. All ships have a name, a ship life state, a current life amount, a current damage buffer amount, an armor class, and a current speed. The life state is either alive or dead; the ship is dead after its current life reaches 0. Each inventory resource has a type and a quantity. A ship also has a positional rotation with x, y, and z components and a velocity direction with x, y, and z components.

A megaship also has a fuel consumption rate, a fuel limit, and an ally combatant ship set. The fuel resource has a maximum value that can be upgraded by players in the game. A megaship has a load flag which it uses to determine if it is being loaded for the first time and must initialize its initial allies.

A combatant is a type of ship. A combatant also has a flag to determine if it is docked and a flag to determine if it is detected by the object it docks to. A combatant has a target velocity with x, y, and z components and a weapon cooldown state. A combatant can be either an ally combatant or a planet guard combatant.

Each weapon has a set of variables used to instantiate weapon emissions: a collision box setting, a damage value, a power level, and a despawn range. A weapon emission represents the actual ‘bullet’ on the map and has a damage value, a power level, and a despawn range. A weapon emission also has a positional rotation with x, y, and z components. The weapon’s values are mutable but a weapon emission is immutable once instantiated. The collision box setting, damage value, and power level are used for damage calculations; the despawn range is used to determine travel distance. The weapon emission also has an origin, which is the combatant that fired it.

An ally combatant has an autostate cooldown, a task state, a list of target enemies, a combat style, a maximum capacity, a maximum damage buffer amount, a maximum life amount, and a maximum speed. The autostate cooldown is used to prevent the ship from changing states automatically when given tasks by the player. The task state is either travel, gather, attack, dock, or idle (this attack is different from the attack state of the combatant because this is the command the player has explicitly given). The combat style is used to indicate modes of attack [ex. Prioritize task, prioritize target, prioritize most recent attacker]. The list of target enemies is a queue of enemy-type combatants that the ally combatant is currently targeting while in an attack state. The ally combatant can set its target destination to be either another map object or a point in space with an x and z coordinate; it keeps track of which version of travel it is engaging in. The ally combatant also has a mode state for whether it is following the current task or engaging in combat.

A planet guard combatant has only a single target enemy at a time, but also has a regeneration rate. A regeneration rate is used to calculate how the planet guard combatant regenerates health over time.

**Assumptions:**

We assume that the entities are User, New Map Template, Saved Game, Map Object, Star, Ship, Planet, Weapon Emission, Resource, Planetary Resource, Megaship, Combatant, Weapon, Ally, and Planet Guard. Star, Planet, Ship, and Weapon Emission are types of Map Object since they will be rendered at a physical location. Combatant and Megaship are types of Ship. Ally and Planet Guard are types of Combatants since they can engage in combat. Weapons are records of allies. Resources are records of Ships and Planetary Resources are records of Planets.

**Entities and Attributes:**

* Ally will have attributes of autostate cooldown, task state, maximum damage buffer amount, a maximum life amount, a mode state, a travel state, and a travel destination (x and z coordinates), as well as a maximum speed and maximum capacity for carrying out the player’s commands to collect resources. It also has attributes of combat style to determine AI behavior in combat~~.~~
* Combatant will have a flag to determine if it is detected, a flag to determine if it is docked, and a target velocity, and a weapon cooldown state.
* Map Object will have attributes of entity type ID, object ID, position (which is a compound attribute consisting of attributes X and Z). The object ID is a primary key because each object ID should be unique. The map object has 4 subclasses in a total disjoint arrangement: star, planet, ship, and weapon emission. This is because a map object must fall into one of these categories but these categories are mutually exclusive entities.
* Megaship will have the attributes of a fuel consumption rate and fuel limit, and first load flag. The exact amount of fuel held by a Megaship is assumed to be stored as a Resource entity.
* Newmap Template will have attributes of template number and version number to track map identity and to track changes to the map. Since the template number is the map’s identity it should be unique (primary key).
* Planet will have attributes of load flag.It will have a relationship state that tracks whether any Planet Guards will attack the player’s Ships. Relationship state is a derived attribute that starts as hostile and becomes allied when the health of all its Planet Guards reaches 0.It also has a life state which is a derived attribute that starts as alive but becomes dead when its Resource quantities all reach 0.
* Planet Guard will have the attribute of a regeneration rate for its health.
* Planetary Resource will have the attributes of current resource count, maximum resource count, regeneration rate, and type**,** which is a weak primary key. A Planetary Resource is a record of a Planet that defines how much of a resource a player can collect from it and how fast that resource will regenerate. A Planetary Resource is a weak entity because it cannot exist without being attached to a Planet, as it is a record of a Planet.
* Resource will have attributes of type and quantity. A Resource is a record of a Map Object that defines how much of a specific resource it is carrying, so it is a weak entity. A Map Object’s set of Resources represents its inventory. Type is a weak primary key.
* Saved Game will have attributes of name, save ID, save parent, and timestamp for identification. Since the save ID should be unique to identify the game save, it is a primary key. It also has a game progress state attribute which is a derived attribute calculated from the health of the megaship and ally ships in a given map save.
* Ship will have attributes of armor class, current damage buffer amount, current life amount, and life state. It will also have a current speed. Finally, the ship will have a name attribute for identification. Life State is a derived attribute calculated from the current life amount of the Ship. When current life reaches 0, the life state becomes dead. It also has x, y, and z rotation components and x, y, and z velocity components.
* Star will have a name for identification on a map.
* User will have attributes of master volume, music volume, effect volume, menu volume, username, and salt and hash, which should be associated with a single player. The username should be unique (primary key) to tell users apart.
* Weapon will have attributes of collision box, damage value, despawn range, and power level for initializing Weapon Emissions. Weapon is a weak entity because it is a record of an Ally
* Weapon Emission will have attributes of damage value, despawn range, and power level for determining damage on impact. It also has x, y, and z rotation components.

**Relationships:**

* Ally Targets Enemy: in which an Ally ship can target one or more Planet Guards at a time. An ally does not need to target a Planet Guard at all times, and not all Planet Guards need to be targeted at once. This relationship has the attribute Queue Number to track the order the ally has targeted particular ships.
* Collects: where the Map Objects Collect Resources to form individual inventories. The Map Object-Resource relationship is 1 Map Object to N Resources because a Map Object can have multiple Resources but a Resource can only belong to one inventory at once.
* Commands: where the Megaship has a retinue of Ally combatants the player can control. A Megaship commands Allies in a 1 Megaship to N Ally relationship because an Ally can only serve one Megaship, but a Megaship can have multiple Allies at its disposal.
* Enemy Targets Ally: where a Planet Guard can target only one ship at a time, but multiple Planet Guards can target the same ship.
* Exists on: where a Planet has a set of Planetary Resources that together compose its resource pool. A Planetary Resource can belong to only 1 Planet and cannot exist independently of a Planet. A Planet might have multiple Planetary Resources.
* Fires: where a Combatant fires Weapon Emissions. There is no limit to the number of Emissions a Combatant can fire but each Weapon Emission can only be spawned once. This relationship is 1:N cardinality because one Combatant may generate many Weapon Emissions but each Weapon Emission can have only one source.
* Initializes: in which a Newmap Template is used to instantiate a new Saved Game instance. A Saved Game can only have one source for Newmap Template initialization, but the same Newmap Template can be used to initialize many Saved Games.
* Lists: where each Saved Game must list all the Map Object entities its state possesses in order to save and load the game. This relationship is 1:N cardinality because there are many Map Objects per one Saved Game but a Map Object can exist in only one game at a time.
* Manipulates: A User Manipulates a Saved Game by either saving or loading a game save. This relationship is 1:N cardinality because one user may have many active saved games but a Saved Game can have only one creator. Users can’t share saved games (so there is one user per saved game).
* Protects: Planet Guards Protect a Planet they are assigned to. This relationship is 1:N because many Planet Guards may protect one Planet but each Planet Guard can only exist on one Planet at a time. Further, Planet Guards have total participation because they can’t exist independently of a Planet.
* Stores: A Newmap Template has one copy of a Saved Game it uses as a master copy to initialize all other Saved Games that rely on the Newmap Template for initialization. This relationship is 1:1 because a Newmap Template represents a container for exactly one Saved Game which it Stores, and it defeats the purpose to uniquely identify Newmap Templates if a Saved Game can belong to multiple such Templates.
* Travel Target: An ally ship may have one travel target, which could be a Star, another Ship, or a Planet; a Star/Ship/Planet can be targeted by multiple ally ships. This relationship is a 1:N due to how a Map Object can be targeted by many Ally Ships, rather than just one, but a Ship cannot target multiple Map Objects, only one specific Map Object.
* Wields: An ally Wields a Weapon that acts as a template for the creation of Weapon Emission objects. This relationship is 1:1 because one allycan only have one Weapon and a Weapon can only exist on one allyat a time. Further, a Weapon cannot exist independently of an ally.

**Additional Assumptions:**

* Entity Type ID is used to indicate the model used in the game and does not have to be unique.
* The weapon emission's origin can be stored as a relationship of who fired it.
* The fact that a weapon was used to instantiate a weapon emission is not important to the relationship structure; further, a weapon emission does not require an origin (weapon emissions can come from God and therefore do not always have a source weapon).
* Ship is a superclass for the megaship and combatant entities in a total disjoint relationship because megaship and combatant are mutually exclusive and a ship must specialize into one of those types.
* A combatant is a total disjoint superclass for planet guards and allies because a combatant must specialize to have a complete set of behaviors and allies and planet guards are mutually exclusive categories.