ChatGPT

Space Base Showdown – Game Design Manual

Introduction

Space Base Showdown is a space-themed strategy game that combines base-building, resource management, and automated squad battles in a cartoon-styled universe. The game is divided into two primary modes: a **Base Management (Preparation) mode** where you construct and upgrade a home base (like a tycoon/idle game), and a **Battle mode** where you deploy troops in 1v1 auto-battles against AI opponents. Your goal is to develop a powerful space base and army, and win battles to gain resources and prestige. By balancing your time between expanding your **home base** and commanding your troops in **auto-battles**, you will progress through the game's ranks and unlock new content. This document provides a comprehensive guide to the game's features, mechanics, and assets so that a developer (or AI agent) can build the game from scratch simply by following these instructions.

Gameplay Overview

In **Space Base Showdown**, players alternate between two interconnected gameplay phases: building up a space base during the **Preparation Phase**, and engaging in **Battle Phase** combat encounters. In the Preparation/Base mode, you will **construct buildings, gather and manage resources, train and upgrade units, and research new technologies** [1]. These activities strengthen your forces for battle. In the Battle mode, you will **face an opponent in a 1v1 match** by deploying up to six of your units onto a battlefield grid; once the battle begins, units fight **autonomously** without further player input, following their AI behaviors. Success in battle grants you more resources and progress, while defeat still provides minor rewards and insight for improvement. The core loop is: **Build -> Upgrade -> Battle -> Earn -> Build more**, creating an engaging cycle of steady base development and exciting combat encounters.

Key Features at a Glance:
- **Base Building & Management:** Develop a home base on an alien world by constructing and upgrading various buildings. Each building serves a unique function (resource production, unit training, research, etc.) and can be upgraded to improve its output or unlock new capabilities [1]. Manage resources produced by your base to expand your facilities and strengthen your army.
- **Resource Collection:** Earn **Galactic Credits** (the primary currency) by winning battles (larger rewards for victories, smaller for losses). Produce other resources (like metals or crystals) via base buildings over time, in idle-game fashion. Use these resources to purchase buildings, unit upgrades, and research new tech.
- **Automated Battles:** Engage in tactical battles where you **pre-select and position your troops** and then watch them fight automatically. All strategic decisions – which units to bring and how to arrange them – are made before combat starts, and then the battle plays out hands-free [2]. Units follow their own AI (e.g. melee units charge forward, ranged units hang back), so victory depends on smart preparation, unit composition, and placement strategy.
- **Progression & Upgrades:** Continuously improve your base and army. Initial upgrades and build times are quick (to get you started), but higher-level upgrades take longer and cost more, introducing an idle progression element. Units start relatively weak but can be upgraded over time through experience or resource investment, increasing their health, damage, and abilities. Research new unit types and abilities to

diversify your army.
- **Space Theme & Aesthetics:** The game features a colorful cartoon-style sci-fi aesthetic. Expect vibrant planet landscapes, quirky futuristic buildings, and charismatic troop designs (aliens, robots, and astronauts) that are visually distinct by role. Ambient spacey music plays during base management, while energetic orchestral/electronic music plays during battles. Fun sound effects (laser blasts, explosions, etc.) accompany actions to bring the battles and base to life.

By the end of this guide, a developer should have all the information needed to implement both the base management systems and the combat systems, as well as the art and audio style, ensuring that launching the game and playing is immediately intuitive.

Base Management Mode (Preparation Phase)

The **Base Management Mode** (Preparation Phase) is where the player builds and administers their home base on a distant planet. This mode plays similarly to a base-building tycoon or idle game. The base serves as the **hub/lobby** of the game – from here, players can access all other features (such as initiating battles, checking units, or starting upgrades). In this mode, time passes and resources accumulate even when the player is not in battle, and upgrades can take real time to complete (idle mechanics). The base mode's main functions are to **generate resources, unlock and upgrade units, and prepare the player for battles**.

Objectives in Base Mode

- **Build and Upgrade Structures:** Create a variety of buildings in your base. Each building provides specific benefits or enables new features (detailed in the next section). Upgrading buildings increases their effectiveness (e.g. more resource output or ability to train higher level units). The ultimate objective is to expand your base's capabilities by unlocking and maxing out all important structures.
- **Resource Management:** Oversee the production and storage of resources. Players must collect produced resources and spend them wisely on upgrades, new buildings, or unit enhancements. Efficient resource management ensures steady progression.
- **Army Preparation:** Train new units or improve existing ones so you have a strong roster for battles. Research technologies that enhance your troops or unlock new troop types. Organize your troops and decide which ones to take into the next battle.
- **Progression & Expansion:** Upgrade your **Command Center (HQ)** to unlock higher tiers of buildings and upgrades. As you advance, more options become available – new building types, higher building levels, additional unit types, and so on. The base will visually grow from a small outpost to a sprawling advanced colony. This progression provides a sense of growth and prepares you for tougher battles ahead.

Resources and Economy

Your interstellar base produces and consumes various resources that fuel your expansion. The economy is designed to support idle accumulation with increasing upgrade costs and wait times at higher levels (typical of incremental/idle games).

- **Galactic Credits (Coins):** This is the primary currency, earned mostly from winning battles and completing missions. Credits are used for **most building construction and upgrade costs**, and also

for training units or researching tech. Winning a battle yields a significant amount of credits, while losing yields a smaller amount as a consolation (e.g. a win might grant 100 credits, a loss 20 credits, scaling up with harder battles) [3]. In base mode, you might also generate a slow trickle of credits via a Trading Post building (for example) to simulate commerce.

- **Metal (Iron Ore):** A common building material produced by mining facilities in your base. Metals are required for structural upgrades and to craft mechanical units or equipment. For example, upgrading the Headquarters or constructing heavy armor for a tank unit might cost metal in addition to credits. Metal is produced over time by an **Ore Mine** building in your base. The player can collect it periodically (the mine will stop accumulating when it reaches capacity).
- **Crystal (Energiun or other space mineral):** A rarer resource representing advanced energy or mystical resource. Crystals are used for high-tech research and magical/spell upgrades. For instance, unlocking a new spell-caster unit or upgrading a shield generator might require crystals. They are produced by a **Crystal Synthesizer** or harvested from special planetary nodes. Like metal, it accumulates idly at a slower rate and must be collected.
- **Energy:** (Optional resource) Energy could represent power needed to run advanced facilities or to launch battles. Some games use an "energy" mechanic to limit battle attempts, but for this first iteration we won't implement an energy/stamina system for battles (players can battle freely). Instead, *Energy* can be a resource produced by a **Power Core/Generator** building and used for certain upgrades (e.g. powering a research lab upgrade).
- **Research Points or Tech Credits:** (Optional) These could be generated by a **Research Lab** over time, indicating accumulated knowledge. They would be spent to unlock new unit types or global upgrades. However, this can be abstracted – instead of a separate resource, we might simply use time + other resources for research tasks.
- **Premium Currency (Star Gems):** *Not implemented in the first iteration.* In the future, a premium currency could be introduced for monetization (purchases with real money). It would be used to speed up build timers or purchase exclusive upgrades. For now, we mention it for completeness but it won't appear in gameplay yet (everything is free and time-based for now).

Resource Generation and Idle Mechanics: Each production building (like the Ore Mine, Crystal Synthesizer, etc.) will generate its resource at a certain rate per minute. Initially, the rates are small but upgrading the building increases production. These buildings have a storage capacity — once they fill up, production halts until the player collects the resources by tapping/clicking the building. The Command Center or separate **Storage Depots** can increase the maximum resource storage for each type. This means the player should log in periodically to collect resources and keep production going, a common idle game mechanic.

Example: An Ore Mine at level 1 might produce 5 metal per minute and store up to 50. Upgrading it to level 2 might increase output to 8 per minute and capacity to 100, but the upgrade could take 5 minutes and cost 100 Credits + 50 metal. Higher levels might take hours or days to upgrade, aligning with the idle game progression (players can either wait or later use premium currency to hurry).

Buildings and Base Structures

Your base starts with a single structure – the **Command Center (Headquarters)** – and a small plot of land. From there, you can construct various buildings. Each building has a specific purpose and provides new capabilities. Buildings can be upgraded to improve their function (e.g. more production, unlocking higher-

tier units, etc.). Below is a list of building types, their functions, and a description of their visual design for artwork.

- **Command Center (Headquarters):** The central building of your base. Upgrading the Command Center unlocks higher tiers of other buildings and expands the base. *Function:* The HQ often gates progression – for example, you cannot upgrade any other building beyond the level of the HQ. It might also passively increase resource storage. *Visual design:* A large futuristic dome or tower, clearly the tallest structure. Since it's a space theme, it could resemble a landed spaceship or a futuristic command tower with antennas and a radar dish. As it upgrades, it becomes more elaborate: e.g., more antennas, glowing windows, additional modules. This building gives the base a sense of presence and is often placed in the center.
- **Ore Mine / Mineral Extractor:** This building produces **Metal** continuously. *Function:* Generates metal resource over time. Upgrading it increases production rate and capacity. *Visual:* A mining facility on the planet surface – for a cartoon style, perhaps a small mine entrance with a drill or laser extractor. It might have a conveyor belt with ore, and perhaps a funny robot miner working. At higher levels, the mine could expand – more drills, maybe a larger crane or multiple extraction points.
- **Crystal Synthesizer / Xeno-Farm:** A producer of **Crystal (Energiun)** resource. *Function:* Slowly generates the special resource used for research and advanced upgrades. *Visual:* A sci-fi structure with perhaps crystals growing in containment, or an array of glowing crystals being cultivated. Could be a dome with crystal formations inside, or a set of pylons drawing energy from the ground. It might glow softly (different colors for different levels, e.g. green at low level, blue/purple at high).
- **Power Core / Energy Reactor:** (Optional resource building) Produces **Energy** resource if used. If not using an explicit energy resource, this building can be omitted in initial implementation. If included: *Function:* generates energy over time, which might be used to power advanced units or buildings. *Visual:* A power plant with a futuristic fusion reactor look – maybe a humming core, coils, and lights. Cartoonish touches like occasional electric sparks could show it's active.
- **Barracks / Training Facility:** This is where basic **infantry units** are trained and upgraded. *Function:* Unlocks and allows training of basic troop types (like Marines, Soldiers). Might allow you to increase the level of those units or produce new units if we had a consumable troop system. Given our design leans towards persistent units, the Barracks could serve as the place to recruit new units (adding them to your roster) or to level them up via training drills when idle. *Visual:* A military-style building adapted to space: perhaps a low bunker with a holographic training yard or a shooting range out front. Could have a symbol like crossed lasers or a target icon. Upgrading it adds more equipment around (satellite dish, extra floor, etc.).
- **Tech Lab / Research Laboratory:** This structure allows **research** of new unit types, abilities, or global upgrades. *Function:* The player can initiate research projects here. For example, researching "Plasma Weapons" could increase all units' attack power, or researching a new unit type "Plasma Mage" unlocks that unit for production. Research tasks cost resources (credits, crystals, etc.) and take time, similar to building upgrades. *Visual:* A high-tech lab building with telescopes, satellite dishes, or magical apparatus depending on tech theme. It might have bubbling energy tubes and a large computer screen. Upgrades could add more satellites or a second story with additional lab equipment. This building's design should scream "science and discovery" – perhaps a mix of futuristic tech and an observatory.
- **Academy / Spell Forge:** (If separate from Tech Lab) This building is dedicated to improving **unit abilities or spells**. *Function:* You can upgrade specific abilities of your units here or unlock new spells that certain units can cast. For instance, increasing the damage of the Mage's fireball or unlocking a

healing spell for the Medic unit. If this is too granular, we can merge this function into the Tech Lab. *Visual:* If included, an academy might look like a futuristic school or a psi-training center with holographic projectors and symbols of knowledge (e.g., a book or a crystal ball icon).

- **Armory / Workshop:** (Optional) A building for crafting or improving equipment for your units. *Function:* Not needed in first iteration unless we add gear systems. If included, players could craft weapons/armor that boost units. This can be skipped for now to reduce complexity.
- **Garage / Vehicle Bay:** (Optional) If the game includes mechanical units or vehicles (tanks, mechs, spaceships), this building would unlock and upgrade those. Could be added in future expansions. *Visual:* A hangar with a garage door where one might see a tank's silhouette. For now, not implemented unless a specific unit type demands it.
- **Storage Depot:** Increases the **storage capacity** for resources like metal and crystal. *Function:* Each storage upgrade lets your base hold more of a resource before it must be spent or collected (important for accumulating for big upgrades). *Visual:* Silos or warehouses – e.g., a metal storage could look like stacked metal crates or silos; a crystal storage might be a secure vault with glowing containers. These should visually correspond to the resource they store (metal vs crystal). Upgrading them makes them larger or adds additional tanks/containers.
- **Defense Towers/Walls:** *Not used in first iteration.* Since currently the base will not be attacked (no PvP base raids yet), defensive structures are not needed. In the future, if base-vs-base combat is introduced, walls, turrets, shields could be built to protect your base. For now, we will exclude defensive buildings.

Building Placement and Base Layout: The base is presented in a **top-down or isometric view** of the planet surface where the player can place buildings on a grid. The style is akin to games like Clash of Clans or Boom Beach (though with a sci-fi cartoon twist). The player can choose locations for each building (within a predefined buildable area). Buildings cannot overlap; the layout is mostly aesthetic in this iteration since there are no enemy attacks. To simplify initial implementation, the base area could have fixed "building slots" or a grid where buildings snap in, avoiding complex pathfinding issues. The **background** of the base view is a planetary landscape: imagine an alien world with a starry sky above. For example, a dusty orange Mars-like terrain or a moon surface with craters. Distant mountains or a ringed planet on the horizon add depth. This background art remains behind the building sprites. The style is colorful and inviting (not dark or scary). Perhaps there are subtle animated touches in the base scene: a flying spaceship in the sky occasionally, or blinking lights on buildings.

Cartoon Art Style Note: All buildings have a stylized, slightly exaggerated look (e.g., oversized features like a giant drill head on the mine, or a comically large antenna on the HQ) to keep the tone light and visually readable. Colors are bright with high contrast outlines to fit a cartoon aesthetic. Each building should be instantly distinguishable by silhouette and color scheme.

Building Upgrade Visuals: When a building is upgraded, its sprite changes to reflect a more advanced look. For instance, the level 1 Command Center might be a small dome, but by level 5 it's a multi-tiered base with multiple domes and antennas. This provides visual feedback of progression. We should prepare 3-5 incremental visual variants for each building corresponding to level ranges (e.g., levels 1-3: small HQ, 4-6: medium HQ with extra modules, 7-10: large HQ fully decked out). Similarly, resource production buildings might get more machinery or larger storage containers as they level up.

Constructing & Upgrading Mechanics: - The player initiates building construction or an upgrade via the base interface. For example, clicking an empty plot or a "Build" button shows a menu of available buildings. Selecting a building will allow the player to place its foundation on the map (drag to a valid location and

confirm). Construction then begins, showing a progress bar or timer. Initially, construction of simple buildings might be instantaneous or only a few seconds (tutorial phase). Higher level upgrades will introduce wait times (minutes or hours). While upgrading, that building might be out of service (e.g., a resource building won't produce during upgrade). - The UI should clearly indicate when a building upgrade is done (e.g., an icon or animation, such as a construction scaffold disappearing). The player can then "complete" it by tapping, and the building's new level functionality becomes active. - For now, allow **one building upgrade at a time** (common in many base builders). Later, we could allow multiple parallel builds if the player acquires additional builders or something similar. But single build queue simplifies initial development. - Optional convenience: a **build/upgrade confirmation dialog** showing cost and time, so players can confirm spending resources.

Unit Training and Management

A key purpose of the base is to allow players to obtain and improve their combat units (troops). In Space Base Showdown, units are persistent (they are not consumed/destroyed permanently when used in battle; if they fall in battle, they are healed and available again afterwards). The emphasis is on **building a roster of various unit types** and upgrading them over time, rather than mass-producing disposable units. Below we outline how players get new units and enhance them, and we describe the different unit types in detail.

Recruiting New Units: Initially, the player will have a small number of basic units (for example, two **Space Marines** as starting troops). To get more units or new types:
- Build the corresponding training structure. For example, building a **Barracks** might unlock the ability to train basic infantry units. Building a **Tech Lab** might allow unlocking high-tech units like a robot or alien.
- Once unlocked, the player can recruit a unit by spending resources (credits, possibly some metal/crystal) in the training building's interface. This might initiate a training time (e.g., 1 minute to train a new Marine). Upon completion, that unit is added to the player's roster. *For simplicity, the first iteration can allow instant training or very short times for new units.*
- There may be a cap on how many units of each type you can have ready (this could be limited by base facilities or just by the fact you only need up to 6 in a battle; we won't impose a hard cap initially beyond logical limits).

Unit Roster and Selection: The game should provide a **Unit Management** screen (or panel within base view) listing all the units the player currently has available. Here you can see each unit's level, stats, and possibly equip them or upgrade them. This UI might be accessible via a "Army" or "Squad" button from the base screen.

Upgrading Units: Units start at Level 1 (or base stats) and can be upgraded to improve their combat performance. There are multiple ways unit upgrades could work; we can implement one or more of the following:
- **Experience and Level-Up:** Units gain XP from battles. For example, each battle participated might give a unit some XP (more if they survive or if you win). After accumulating enough XP, a unit's level increases, raising its stats (health, damage, etc.). This happens automatically when thresholds are reached. We can show an XP bar for each unit in the roster UI.
- **Training Upgrades:** The player can send a unit to train (for example, in the Training Facility building) to increase its level or rank. This could cost some resources and time. For example, upgrading a Marine from level 2 to 3 might cost 50 Credits and take 5 minutes. This is an alternative or supplement to XP; depending on design choice, we could allow both XP leveling and direct upgrading by spending resources (one

representing combat experience, the other formal training).
- **Research & Tech Buffs:** Researching technologies in the Tech Lab can improve entire categories of units. For instance, a research project "Ballistic Weapons Upgrade I" could increase the damage of all bullet-firing units by +10%. This provides a global boost and gives another progression avenue. Another example: researching "Advanced Armor Plating" might increase all units' shield or health. These upgrades affect all current and future units of those categories.
- **Unit Rank Evolution:** Perhaps certain units can be "ranked up" when they hit a max level, resetting them to a stronger version (common in RPG gacha games). For simplicity, we might skip this for now, but we can note it as a future addition: e.g., a Marine at level 10 can be promoted to "Veteran Marine" with improved stats and a new look.

Any upgrade or training that takes time can use the same timer mechanics as building upgrades, reinforcing the idle aspect (short times for low levels, longer for high).

Unit Attributes: Each unit has a set of stats that determine their effectiveness in battle: - **Health (HP):** How much damage the unit can take before being defeated. Some units also have **Shield** in addition to health; shields regenerate between battles (or even during battle for advanced units), whereas health might not regenerate in-battle. Shields effectively act as a secondary HP bar that is usually smaller but can refresh. Early on, shields are weak, but research can improve them. - **Damage (Attack Strength):** How much damage the unit deals with each hit (or per second for continuous damage). This can be a base damage value. Some units might do area damage (specified in their ability description) while most do single-target damage. - **Attack Speed (Rate):** How fast the unit attacks. This could be expressed as "attacks per second" or a cooldown (e.g., one attack every 2 seconds). A higher attack speed means more frequent attacks and thus higher DPS (damage per second). - **Range:** Especially for ranged units, how far they can attack from. Melee units have range 1 (adjacent tile or short melee distance), whereas a sniper might have a long range (they can shoot enemies from many tiles away). Range affects their behavior – ranged units will stop at a distance to shoot, while melee must base themselves adjacent to enemies. - **Movement Speed:** How fast the unit moves across the battlefield grid. A faster unit can close distance or reposition quicker. Typically measured in tiles per second. For example, a light assassin might move very fast, whereas a heavy tank unit moves slowly. - **Special Ability:** Many units have a unique trait or ability that sets them apart. For example, a medic unit might periodically heal the ally with lowest health, a flamethrower unit might do splash damage in a cone in front of it, or a unit might have a one-time shield that activates when health drops low. These abilities add depth and can be upgraded through research or unit leveling.

Unit Types and Roles

There will be a variety of unit types for the player to use, each fulfilling a role in combat. Drawing inspiration from similar games (like auto-battlers and RPGs), units can be categorized by **roles such as frontline/ tanks, damage dealers (DPS), support, etc.** [4] . Below is a list of example unit types we plan for the initial game, along with their role, a description of their behavior, and the artwork concept for their sprite design. (All unit designs are done in a cartoony, stylized sci-fi manner.)

1. **Space Marine** – *Role:* **Frontline Fighter / Melee DPS**.
2. **Description:** A basic human infantry unit equipped with a laser rifle (for short range) and light armor. Space Marines are versatile and form the backbone of your forces. They deal moderate damage and can take moderate damage. In battle, Marines will advance forward, then shoot at

enemies once in range. They prefer to stop and fire from a short distance rather than engaging in hand-to-hand combat, but if an enemy closes in melee they can fight up close with a rifle butt strike.

3. **Stats:** Medium health, no shields initially, moderate attack damage, short attack range (3-4 tiles), decent attack speed, average movement speed. No special ability at level 1, but through upgrades they might gain a grenade attack (small area damage on a cooldown).

4. **Sprite Art:** A cartoonish astronaut-soldier wearing a white and blue space suit with a visored helmet. They carry a blaster rifle that emits colorful laser beams. They might have a jetpack (just decorative initially). The design should be friendly but clearly a soldier. They could have a confident stance. When attacking, an animation of firing the rifle with a laser beam effect should be shown. When hit, maybe a small spark or flash on their armor. When defeated, they could fall over dramatically with a little "cartoon dust cloud."

5. **Star Defender** – *Role:* **Tank (Frontline Defender)**.

6. **Description:** A heavy armored unit whose job is to protect other units by absorbing damage. The Star Defender carries a large energy shield and a close-combat weapon (like an energy sword or hammer). In battle, this unit marches at the forefront, drawing enemy fire. They have high health and an energy shield that soaks damage until it breaks. They do only modest damage themselves. Their fighting style is to engage the nearest enemy in melee and hold them off.
7. **Stats:** High health, plus a shield that refreshes slightly between battles (e.g., 50 shield points that regenerate after each battle). Low attack damage, melee range (1 tile), slow attack speed (big swings), slow movement (heavy armor).
8. **Special:** Perhaps a passive ability – e.g., "Shield Wall" that briefly reduces all damage they take by 50% when their health drops below 20% (triggered once per battle). This makes them even harder to kill.

9. **Sprite Art:** A big, burly character in bulky sci-fi armor. Exaggerate the size difference compared to a Marine. The Defender holds a **large rectangular energy shield** in one hand (glowing edges) and a melee weapon in the other (like a futuristic mace or sword). The armor could be colored in strong, defensive colors (deep blue or grey), with visible plating. Their helmet might have a distinct crest or faceplate. The cartoon style lets us make them slightly humorous (maybe the shield is almost as big as the character). Attack animation: a slow swing or bash with the shield. When hit, the shield flashes to indicate it absorbed damage.

10. **Cosmic Sniper** – *Role:* **Ranged DPS (Backline)**.

11. **Description:** A specialist in dealing high damage from long range. The Cosmic Sniper is fragile but has a powerful railgun or sniper rifle that can pick off enemies before they get close. In battle, the sniper will stay near the back of your formation and target the closest or weakest enemy within a very long range. They try to keep distance; if enemies approach, they will even move backwards (if possible) to maintain range.
12. **Stats:** Low health, maybe no shield or a very small one. Very high damage per shot (can significantly hurt or one-shot weaker targets), very slow attack speed (long reload between shots), very long range (can hit almost across the battlefield if unobstructed). Very slow movement (snipers tend to be immobile while aiming).

13. **Special:** High accuracy – their shots never miss. Possibly an ability: “Camouflage” – the first time an enemy comes within 2 tiles, the sniper cloaks and moves 2 tiles backward (retreating once per battle). This helps them survive a bit longer against charging enemies.

14. **Sprite Art:** A sleek-looking character in a stealth suit or light armor. Perhaps an alien or human with a high-tech sniper rifle that has a long barrel and a scope with a glow. The sniper could have a cloak or cape that suggests camouflage. They might have a single eye visor targeting system. Since it’s cartoon style, we might give them a dramatic pose when aiming (e.g., kneeling on one knee with the rifle). Firing animation: a bright laser or projectile that crosses the battlefield quickly with a trail, maybe a small explosion on impact.

15. **Nova Mage** – *Role:* **Spellcaster (Ranged AoE / Support)**.

16. **Description:** A unit that uses “space magic” or advanced tech to cast damaging spells or support the team. The Nova Mage stays at the rear and launches elemental energy attacks. For example, they might cast a **plasma bolt** that explodes and deals area damage to multiple enemies, or a **gravity field** that slows enemies. They prefer to stay as far back as possible. They are vulnerable if enemies reach them.
17. **Stats:** Low to medium health, some shield (mages might have personal shields for protection). Moderate movement speed (they float or glide?). Attack: moderate damage but hits multiple enemies in a small radius (say a fireball that splashes). Attack speed is slow (casting takes time, e.g., one spell every 3 seconds). Range is long (not as long as sniper, but enough to stay safely behind frontlines).
18. **Special:** Area of Effect attack – their basic attack hits 2-3 enemies if they are clustered. Additionally, perhaps a secondary effect: e.g., “Cosmic Flame” continues to burn targets for a short duration (damage-over-time) or “Gravity Pulse” slows those hit. We can design one to make the mage unique. Upgrades can increase the radius or effect.

19. **Sprite Art:** A robed figure, possibly alien or human, wielding a staff or orb. This is a chance to have an alien species in the roster – e.g., a grey alien or a glowing energy being – to emphasize the fantasy sci-fi mix. The mage could float slightly above the ground to show mystical power. They have glowing eyes or a magical aura. The staff or hands glow when casting. Visual effects for spells: For a plasma bolt, a bright purple orb is launched; on impact it creates a colorful explosion with stars/sparkles. The cartoon style could give spells an exaggerated look (big, bubbly explosion).

20. **Medic Drone** – *Role:* **Support Healer**.

21. **Description:** A small flying drone (or a medic unit) whose primary job is to heal your units during battle. The Medic Drone hovers behind your combat units and automatically restores health to injured allies. It has no (or very weak) attack of its own. This unit can be crucial for keeping your frontline alive, but it’s defenseless if enemies get to it.
22. **Stats:** Low health, no shield or a tiny shield. It should avoid combat. Healing power: e.g., heals an ally for X HP every few seconds. Range: can heal from moderate distance (stays behind frontline but within, say, 4-5 tiles of the unit it heals). Movement: fairly quick (it zips around to reach damaged allies).
23. **Special:** Healing beam – targets the ally with lowest health automatically. Possibly an area heal ability on a longer cooldown (like an emergency burst heal to all allies in range once per battle). It does not revive dead units (once a unit’s HP hits 0 in battle, they’re out for that battle).

24. **Sprite Art:** A cute hovering robot drone with medical decals (a little red cross symbol or blue plus sign). It might have little rotor blades or anti-gravity pods. When healing, a beam or particle effect links the drone to the ally (e.g., a green beam or a shower of sparkles) to indicate healing. The drone might have an orb-like body with blinking lights. For a cartoon vibe, we could give it "eyes" (like a single camera lens that tilts to show emotion). It should look non-threatening and helpful.

25. **Assault Bot** – *Role:* **Melee Bruiser / Damage Dealer**.

26. **Description:** A robotic unit that charges into melee and deals high damage with built-in weapons (like saws or fists). It's tougher than a human marine and hits harder, but has less tactical flexibility. The Assault Bot is great for punching through enemy lines. In battle, it barrels straight toward the nearest foe. It might even ignore smaller enemies to hit a high-value target (depending on AI programming).
27. **Stats:** High health (not as much as Defender, but solid), maybe moderate shield. High melee damage output, moderate attack speed (faster than Defender's swings, slower than a Marine's rifle). Movement speed: medium. No ranged capability – must reach target to hit.
28. **Special:** Possibly "Charge" – at the start of battle or when it spots an enemy, it accelerates, closing distance quickly (a one-time burst of speed). This gets it into combat fast. It could also have a small area attack (its swings hit two enemies if they stand close).
29. **Sprite Art:** A chunky bipedal robot, somewhat reminiscent of a cartoon mech. Maybe about the same size as the Defender. It has big arms with built-in weapons (e.g., a jackhammer arm or an energy blade). Designed in a friendly cartoon style, perhaps with a dome head or a single eye sensor on the chest. Its color scheme could be industrial (yellow/black stripes) or a bright color like orange. Animation: walking with heavy stomps; attack: a double-fist slam or a spinning attack.

(The above are examples; we can adjust the number of unit types as needed. The idea is to have a diverse roster covering basic roles: at least one tank, one general fighter, one ranged, one support. More units can be added in future updates, keeping the game fresh.)

Unit Role Emphasis: Each unit's AI behavior is tied to its role. For instance, *Frontliner/Tank units* will position themselves at the front of your formation and are coded to **engage the enemy head-on**, attracting fire and protecting allies [4]. *Ranged units* will try to maintain distance, attacking from afar, and might reposition to avoid melee. *Support units* will hang back and focus on their support tasks (healing or buffing) instead of seeking enemies. Designing AI in this way ensures that, just as in other strategy auto-battlers, you must think about having a **balanced formation with strong frontliners, consistent ranged DPS, and possibly at least one support** [5]. The game will encourage experimenting with formations and combinations – for example, pairing a Medic Drone with a Defender can make that Defender much harder to kill, or using a Nova Mage behind a Tank can deal area damage while the Tank holds enemies in place. These synergies make the preparation phase meaningful.

Base Mode Controls & UI

The base management interface should be intuitive and accessible, especially given the cartoon style (which implies a broad audience, including casual players). Key controls and UI elements in base mode:

- **Camera View:** The base is shown from a semi-top-down perspective. It likely fits on one screen at first, and can scroll if the base gets larger. The player can click and drag (on web, using mouse or touch) to pan around the base if needed. Zoom could be optional but not necessary initially.
- **Building Menu:** A button (for example, "Build" or a hammer/wrench icon) opens the construction menu. This shows icons of buildings available to construct, with descriptions, costs, and requirements. Buildings that are locked (e.g., require HQ level X) should be shown with a lock icon and info on how to unlock.
- **Constructing a Building:** After selecting a building from the menu, the player can move the ghost outline of the building over the base area to choose a placement. Valid placement squares highlight in green, invalid in red. Click to confirm placement. (Alternatively, simpler: have predetermined slots where the building will automatically go when built, to skip free placement – e.g., a fixed base layout progression. But free placement is more tycoon-like and engaging.) Once placed, if there's a build time, a progress bar or timer appears on it. If build is instant (for early buildings), it just appears.
- **Upgrading Buildings:** Clicking on an existing building brings up a small panel with building details: current level, what it produces or its effect, current output, etc., and an "Upgrade" button (if upgrade is available). The upgrade button should show cost and time. If the player has enough resources, clicking it starts the upgrade (with a confirmation if desired). The building might show scaffolding or a construction animation during the upgrade countdown. Only one upgrade can run at a time initially. Possibly show a small timer icon on the building and maybe a global timer at top UI for the current upgrade.
- **Collecting Resources:** Resource-producing buildings (mine, synthesizer) will show an icon or indicator when resources are available to collect (for example, a floating resource icon or the building itself visually fills up – e.g., the mine might show a pile of ore when full). The player clicks the building to collect, which adds the resources to their totals (displayed in the HUD). This resets the building to empty so it can produce more. The top of the screen UI should have counters for major resources (Credits, Metal, Crystal, etc.) so the player always sees their current amounts.
- **Unit Management Screen:** Accessible via an "Army" button or perhaps clicking the Barracks/Training Facility. Here, the UI would list units, allow upgrades, and possibly allow you to select which ones are currently active or to form a battle squad. For this iteration, you can assume all units you have are available to pick for battles, but we could limit active squad size. In any case, the unit screen shows each unit's icon, name, level, HP, damage, etc. It also might let you initiate an upgrade (like a "Train" button on a unit to spend resources to level it up if allowed). If new units can be trained, this screen or the respective building would have a "Recruit Unit" list (for unlocked types) where you can train a fresh level 1 unit.
- **Navigation:** Since the base is the main hub, ensure there are clear ways to enter other parts of the game:
- A "Battle" button or panel (maybe the Command Center or a "Mission Control" building can serve this function) that takes you to the battle setup interface.
- Possibly a "Research" button (or tap the Tech Lab) to open research menu.
- Maybe a "Quests" or "Missions" panel if we have any mission system (not required for first version, but could be daily tasks or achievements).
- A menu for settings, help, etc., typically in a corner.

The UI style should remain consistent with the cartoon theme: vibrant colors, slightly playful icons (e.g., resource icons that are easily recognizable and a bit exaggerated – a coin stack for Credits, a chunk of ore for Metal, a crystal icon for Crystal resource). Use clear font for numbers and avoid clutter. Animations for feedback (like resource icons popping up and flying to the resource counter when collected) will make the base mode feel alive.

Battle Mode (Combat Phase)

The Battle Mode is where the player tests their squad of units against an enemy squad. Battles in **Space Base Showdown** are designed to be **fast-paced, automated engagements** that emphasize strategic planning over twitch reflexes. The player's role is to **select up to 6 units and position them on the battlefield grid** during a preparation stage. Once the battle begins, units will fight on their own according to their AI routines. The outcome is determined by the composition of units and their positioning (and of course their upgrade levels and synergy). This mode provides excitement and rewards which feed back into the base-building progression.

Battle Preparation and Match Setup

To initiate a battle, the player goes through the following steps:

1. **Entering Battle Mode:** From the base, clicking the **"Battle"** button (or perhaps an in-world building like a **Starport** or **Command Center** with a "Deploy" option) will open the battle preparation interface. This can be presented as a "mission briefing" or "match lobby" screen. At this stage (in the single-player iteration), the player will face an **AI-controlled enemy team**. Later, this flow will connect to online matchmaking, but for now it will either randomly generate or pre-define AI opponents of roughly appropriate difficulty.
2. **Choosing an Enemy/Level:** Initially, the game might offer a campaign-like progression of AI opponents (for example, a sequence of sectors or arenas with increasing difficulty). Alternatively, a simple **"Find Match"** button can pit the player against a suitably ranked AI opponent. We will implement a basic system where the AI's strength is scaled to the player's current battle rating or base progress, ensuring a fair challenge. For example, early on you fight an AI with just 2-3 weak units, but later they field the full 6 units with higher levels. *(This mimics how many games start with easy AI/bots to teach basics before moving to real PvP) – indeed, many auto-battler games feature mostly bots in early ranks* [6] *.*
3. The battle preparation screen can show the enemy team composition (if we want to allow the player to see what they're up against) or it could be partially hidden for surprise. For now, it's fine to show the enemy's units so the player can strategize.
4. **Squad Selection:** The player must select which units from their roster to bring into battle. They can choose up to **6 units** for a full squad (if they have that many available). Early in the game, the player might have fewer units, so they take whatever they have (e.g., 3 units). The interface can show the player's roster on a sidebar or bottom of the screen – the player then drags and drops the desired unit icons into a **deployment bar** or directly onto the battlefield. We should display the limit (e.g., "3/6 units selected"). We may also allow multiple of the same unit type if the player owns multiples (for instance, two Marines can both be deployed if you have two in your roster). If we treat units like unique heroes, we might limit one of each type; but since we allowed training of new units, the player could indeed have two Marines leveled separately. For simplicity, treat each unit in roster as a distinct entity; duplicates are allowed if the player recruited more than one of that type.

5. **Formation Placement:** After choosing which units to deploy, the player places them on their side of the battlefield grid. The battlefield is a grid of **6 rows (depth) by 18 columns (width)**. Visually, it's a side-view 2D battlefield (imagine looking from the side, like a platformer, but with multiple lanes stacked vertically). We can represent it as a grid of tiles: 6 rows (top to bottom), 18 columns (left to right). The **left side** of the grid is the player's deployment zone, and the **right side** is the enemy's. Specifically, the player can place units only within the leftmost 5 columns *of each of the 6 rows* (a 6x5 area), and the opponent will place within the rightmost 5 columns on their side. That leaves about 8 columns of neutral space in the middle where units will meet and fight. This setup ensures a distance between the two starting formations, giving ranged units time to shoot as armies approach.
6. The UI should display the grid (you can show faint grid lines or at least distinct row positions). The player drags each selected unit's icon onto a specific tile in the left-side deployment zone. Once placed, the unit's sprite appears on that tile. The player should consider unit roles when placing: e.g., put tanks in front (rightmost column of your zone, closest to enemy) and fragile units behind (leftmost column of your zone, farther from enemy). Also, units can be placed in different rows to create a formation spread (for example, a tank in row 3 column 5, a tank in row 4 column 5, and behind each a ranged unit in column 3 of same rows, etc.). This positioning will affect who engages whom when the battle starts.
7. The player can reposition units freely during this prep phase: dragging a unit to a different tile, or removing it back to roster (perhaps by dragging it off or clicking to remove). This allows experimentation. We might also include a "suggest formation" button for beginners, but not necessary.
8. If the enemy team is known, the player might see their units already placed on the rightmost side (grayed out or just preview). If we want more Fog of War, we could hide enemy placement until battle starts, but seeing them allows strategic counter-placement.
9. **Begin Battle:** When ready, the player hits a **"Start Battle"** button. If needed, a confirmation sound or countdown ("3, 2, 1, Go!") plays, then the battle commences. At this point, the player can no longer control units; they become observers watching the outcome of their setup.

Battlefield Layout and Visuals

The battle takes place on a single static screen representing the combat area. Since we use a side-view perspective (like a 2D side-scrolling game perspective, not top-down), the units will appear as 2D sprites on this battlefield, moving left or right (and possibly up/down within the lanes). **Background Art:** The battleground has a distinct background separate from the base view – for instance, an open alien battlefield or arena. It could be an open field on the same planet as the base or perhaps a futuristic arena designed for skirmishes. Given the "football field from bleachers" analogy, one could imagine a side-on view of a flat plain that extends to the horizon. Some background ideas: - A rocky alien plain with distant mountains or craters. The sky above could show stars or a nebula (since it's space-themed). Perhaps a giant planet or moon looms on the horizon, giving a dramatic sci-fi effect. - Alternatively, a space station interior or an arena: metallic floor with force-field edges, and space visible beyond a dome. But this might be more complex to illustrate. Initially, the outdoor planetary battlefield might be easier and more visually clear. - The battlefield ground itself can be a mostly flat line where units stand, with slight indication of the 6 rows (maybe subtle markings or different ground textures for each row lane). We can depict the lanes like stripes on a sports field, or slight depressions, to show that units in different rows do not collide unless they move into the same row. - Environmental details: It's nice to have some animated elements to avoid a static look – e.g., drifting clouds or animated flora/fauna in background. But nothing that distracts from the combat.

Grid and Movement: Although the battle uses a 6x18 grid concept for placement and movement, we don't need to display the grid lines overtly (maybe only during placement as highlights). Units will move in continuous fashion (tile to tile or pixel by pixel) towards each other rather than hopping tile by tile, to appear smooth. However, logically they occupy a certain tile at any given time for interaction. There are 6 possible vertical positions (rows) that correspond to lanes. Typically, units will stick to their row as they move horizontally toward enemies, unless they have AI to change lanes (not in this design, units generally stay in their lane unless chasing). So it's like 6 parallel lanes of possible movement, but units can attack diagonally or move up/down if needed to reach an enemy (for simplicity, they might only move vertically if no enemy is in their lane and there's one in an adjacent lane within some range).

Before getting into battle flow, let's outline some **Battlefield UI elements**: - Health bars: Each unit (player's and enemy's) should have a small health bar displayed either above or below them, so the player can see how close each is to dying. Shields could be represented as a second bar (perhaps layered or a different color, like a blue bar that overlays part of the health bar and depletes first). This allows the player to follow the battle's progress. - Maybe icons above units indicating special status (like a small shield icon if a Defender's damage reduction triggers, or a stun icon if a unit is stunned by an ability). - A battle timer if we implement a time limit (e.g., 60 seconds countdown). Initially, battles likely end quickly, but a timer (perhaps 2 minutes default) ensures no infinite stalemates. If time runs out, we could declare the side with more units still alive the winner, or a draw (but draws could just be treated as a loss for both or something similar). - Speed controls: In some idle battles, players can toggle speed (1x, 2x) or pause to watch carefully. For the first version, we can keep it simple at 1x speed, no pausing (except maybe if debugging). Possibly later add a speed-up button for impatient players. - A "Forfeit" or "Retreat" button: If a player sees they will lose, they might quit early. This would count as a loss and end the battle immediately. - No player input to cast spells in this version (all unit abilities are automatic). In future, we could consider a system where the player has a couple of commander skills to intervene, but not now (keeping fully auto).

Battle Flow and Unit AI Behavior

Once the battle starts, both sides' units will begin executing their AI scripts. The combat is real-time and simultaneous. Below is how the battle might proceed:

1. **Initial Advance:** At the moment battle commences, units typically **move forward** from their starting positions. Player units move to the right, enemy units move to the left. They continue this until they encounter an enemy within their attack range or blocking their path. For example, melee units will close the distance all the way until they are adjacent (or collide with an enemy), while ranged units will stop earlier once an enemy is within shooting range.
2. **Target Acquisition:** Each unit will pick a target according to its role and situation:
3. Generally, a unit attacks the **closest enemy** (in terms of straight-line distance) that it can "see" in its lane. If an enemy is directly ahead in the same row, that's the obvious target. If multiple enemies are in range, some prioritization can be set (for example, DPS units might target the lowest health enemy first to eliminate it, or target whoever is frontmost).
4. If no enemy is in the same row/lane, ranged units might shoot at an adjacent lane if within slight angle, but melee units would possibly move vertically to chase if the fight is happening in a different lane. To keep things straightforward: units primarily focus on enemies in their own row. If one side has staggered formation (e.g., enemy in row 1, but player has no unit in row 1, the enemy will move freely until a player unit from another row comes within some proximity).

5. Support units (like Medic) target friendly units for healing rather than enemies. So their “target” is the ally with lowest HP within heal range. They will follow that ally loosely to stay in range.
6. **Combat Engagement:** When a unit has an enemy within attack range and their attack cooldown is ready, they will perform an **attack**.
7. Melee attack: if a melee unit like the Defender reaches an enemy, it swings its sword, dealing damage to that enemy’s HP (or shield first, then HP). If it kills the enemy, that enemy’s sprite plays a death animation and is removed from the field. If not, the enemy’s health bar goes down. Melee units will continue to chase and attack as cooldown allows.
8. Ranged attack: if a ranged unit has a target in range, it will stop moving and shoot. For example, a Marine will stop a few tiles away and fire bursts from their rifle. Each shot (or burst) deals damage. They continue this as long as the enemy remains in range and alive. If the enemy moves out of range (or is dead), the ranged unit will move forward again (until another enemy enters range).
9. Area attacks: units like Nova Mage, if they attack and cause splash damage, will damage multiple enemies at once. The implementation can find enemies within X radius of the impact point and apply damage to each.
10. Attack animations and effects: Show muzzle flashes, laser beams, explosion effects as appropriate to make combat visually satisfying.
11. Attack cooldown/Speed: Each unit has an attack interval (e.g., Marine might shoot every 1 second, Sniper every 3 seconds). Internally, after an attack, a timer runs until the next attack is possible. The unit will continue other behavior (moving or just waiting) during the recharge.
12. **Movement and Spacing:** Units will try not to overlap. If a frontline from each side meet in the same tile or area, they essentially engage in melee there. Other units behind will stop at some respectful distance (or crowd around if in same lane). It might be simplest to allow multiple units to stack in a tile for coding, but visually they should appear slightly offset rather than all on top of each other – consider separating rows for this to avoid overlap issues. Each row is distinct, so units in the same row could bump into each other. Perhaps treat each tile as capable of holding one unit per side at a time. If two of your units would end up in the same tile (like both rushing to fight one enemy), maybe one stands just behind the other. This might not need explicit coding beyond maybe limiting movement if space occupied. But we can be forgiving in visuals.
13. Vertical movement: If an enemy is directly above or below in a neighboring lane and no enemies in the current lane, some AI could decide to move to that lane to engage. However, in most auto-battlers, units stick to their lane except maybe flying units or certain agile ones. For now, we can keep it simple: units do not change rows once battle starts, they move straight horizontally. If one side had no unit in a lane, the opposite unit in that lane goes right through and can flank others from behind. This adds a tactical aspect to initial placement – you don't want to leave a lane completely empty if the enemy has a unit there, or it can go hit your backline. That being said, we could allow units that reach the far end to then move vertically towards nearest fight, but that might complicate things. To balance, the AI enemy will usually mirror player’s placement so such mismatches are less frequent early on.
14. **Autonomous Abilities:** Units with special abilities (like Medic’s heal or Defender’s damage reduction) will trigger those automatically based on conditions:
15. Timed abilities (like a heal every 5 seconds) just run on a loop.
16. Reactive abilities (like shield on low HP) trigger when condition is met.
17. These do not require player input. The game should handle the logic and effects (e.g., when Medic’s heal is off cooldown, find ally with lowest HP and heal them X amount, show heal animation).
18. **Battle Progress and Resolution:** The battle continues in real-time with units attacking, taking damage, possibly moving if someone in front dies, etc. Eventually, one side will lose all its units. At that moment, the battle ends immediately.

19. **Victory:** If the player's units eliminate all enemy units, the player wins. Victory could be accompanied by a celebratory sound, a "Victory!" text on screen, and perhaps fireworks or confetti in a playful way. The battle ends.
20. **Defeat:** If the player's units are all eliminated first, the enemy wins and the player loses the battle. Show a "Defeat" text and a sad sound. Possibly still highlight if the enemy had surviving units ("Enemy units remaining: 2" etc.) so player can consider what went wrong.
21. **Draw:** If by some rare chance both last units die at the same time or time runs out with equal outcome, it could be a draw. We can either treat a draw as a loss or just neutral. Draws will be rare, so not a big focus.

22. If a **time limit** is in place and it hits zero, determine winner by remaining units (who has more alive) or remaining total HP as a tiebreaker. Or simply call it a draw. We will likely set a time limit like 90 seconds to prevent endless battles (some idle games do ~1-2 minute battles) [7]. If needed, later implement a sudden death (units take increasing damage over time to force an end).

23. **Post-Battle Rewards and Results:** After the battle outcome, a **results screen** is shown:

24. Rewards gained: **Credits** earned (e.g., "+100 Credits" with an icon), and possibly some **resources or loot**. For example, winning might also yield a small amount of Metal or a chance at a bonus item. We can keep it simple: each win yields a fixed credit reward plus maybe one loot crate (which could contain a bit of common resource or a unit shard if we had that system). Losing yields maybe 1/4 of the credits of a win and no loot.
25. If we have a concept of **Battle Score or Rank Points**, display the change: e.g., "+10 Battle Rating" on win. If lost, maybe "-5 Rating" (or none for PvE; since these are AI, we might not penalize rating for losing to AI, or we could for the sake of matching difficulty).
26. Show statistics: e.g., list each unit of the player, how much damage they dealt, how much they took, kills, healing done by medics, etc. This info can help the player adjust strategy (and it's fun to see).

27. An "OK" or "Continue" button to return to base mode when ready.

28. **Return to Base:** The player is taken back to their base screen, where they can spend their newly earned resources on upgrades, or queue up the next battle.

Matchmaking and AI Opponents

For the initial single-player experience, all battles are against AI-controlled enemies. We simulate the feel of PvP by using AI squads that follow similar rules. Here's how we handle this: - **AI Squad Composition:** We will design a set of enemy squads of increasing difficulty. Each squad will have up to 6 units (fewer in early fights) with certain levels. For example: - First battle (tutorial): Enemy has 2 basic units (like Alien Grunts) at low level. - After a few easy ones, enemy squads ramp up to 3 units, 4 units, etc., eventually 6 units that are well-upgraded. Their unit types can mirror what the player has seen/unlocked, plus maybe a few unique enemy-only types for flavor (like alien creatures). - The AI units behave using the same mechanics as player units. We just might give them different skins (say the enemy are aliens vs the player's more human units, just to differentiate visually). - **Scaling Difficulty:** Each time the player wins, their **Battle Score** (or "Arena Trophy count") increases. This is similar to a ranking or matchmaking rating. The game can use this to select the next opponent. Essentially, we could have brackets (0-100 rating = easy bots, 100-200 = medium, etc. or in terms of "Tier 1, Tier 2"...). Winning moves you to fight a tougher AI next; losing might keep you at the same level until you improve. Over time, AI units level up or new unit types appear, providing fresh

challenges. This mimics multiplayer matchmaking where you face stronger players as you climb [8]. - **No actual networking in this version:** We will **not implement real-time multiplayer yet**. However, we design the system such that it could be expanded: i.e., the battle mode is essentially agnostic to whether the opponent is AI or a real player's squad. For now, the "matchmaking" is just picking an AI config. In future, an online matchmaking service (like AWS GameLift) would pair two online players of similar Battle Score, then each client would send their formation to a server to simulate the battle. To prepare for that, ensure **battle resolution code can run deterministically given two sets of units and a random seed** (to avoid desync). Also design the architecture so that all combat rules are known and can be run on a server or locally in the same way. This will make integration with online multiplayer seamless later.

Combat Example Scenario

To illustrate a typical battle: The player deploys 1 Defender (front row center), 2 Marines (one in front row slightly behind Defender, one in second row), 1 Sniper (back row), and 1 Medic (back row behind Defender). The enemy AI has perhaps 1 big melee alien, 2 alien archers, and 1 shaman healer. When battle starts: - The Defender marches forward down the center lane. The Marines follow in their lanes (one behind Defender, one in row 2). The Sniper and Medic stay back. - The enemy melee alien charges straight toward the Defender. The archers hang back on their side shooting arrows at the Defender. The shaman heals the alien melee. - The Sniper from afar shoots one of the archers, taking it out after a couple of hits. The Marines open fire as soon as in range. - The Defender and alien melee clash in the middle; Defender's high health holds the alien while Marines and Sniper deal damage. The Medic heals the Defender periodically to counter some archer damage. - Eventually the alien melee dies (focused by many units). The archers, now also targeted by the Marines, get eliminated. The enemy shaman is last; with no frontline, it gets quickly dispatched by the player's units. The player wins with perhaps all units surviving thanks to the Medic's healing and the strong formation. - The result screen then shows maybe "Victory! Units lost: 0, Credits earned: 100, Metal earned: 20" etc., and the player's Battle Score increases. They return to base to perhaps unlock a new unit type with the resources.

This scenario shows how **formation and unit mix matter:** had the player not brought a Medic, maybe the Defender would have died earlier and the battle could swing the other way. If the player had no Defender, the alien melee might have reached the squishier units and caused chaos. Thus, strategy in unit choice and placement is rewarded.

Battle Mode Controls & Observing

During the battle itself, the player is mostly an observer. However, we can provide some controls to make watching or controlling the flow convenient: - **Camera:** The camera can be fixed showing the entire 18-column battlefield. Units will be relatively small if we show all 18 columns on a typical screen, but given 6 rows, it should be manageable. If needed, we could have the camera follow the action (scroll as units move), but that might not be necessary if we fit it. Probably better to design sprites and field size such that everything fits without scrolling, to avoid the player missing something. - **Speed Toggle:** (Optional for first version) A button to speed up the battle (2x or 3x) for players who don't want to wait especially if they're confident. Also possibly a pause for taking a screenshot or analyzing, but not needed in normal play. - **No input on units:** There is intentionally no micro-management of units mid-battle (no ability activation by click, etc.) in this design, aligning with auto-battler principles [2]. The player's interaction is limited to watching. This design choice emphasizes the planning stage; as one analysis notes, *all strategic decisions are made prior to the match, and the actual battle is entirely automated* [2]. This can feel rewarding as a

commander – you get to see your strategy play out like watching a simulation, rather than twitchy control. - **Retreat:** If implemented, a “Retreat” button could allow the player to exit a battle early, counting as a loss. Useful if you realize a battle is unwinnable and you don’t want to wait. This is optional; players could also just watch until defeat.

AI Behavior Details (for coding)

To implement the battle logic, here are some specifics for the developer: - Each unit will have a simple state machine: e.g., **State: Moving** (towards enemy) or **State: Attacking** (enemy in range). Support units might have **State: Healing**. - We can update the battle in discrete time steps (e.g., 10 or 20 updates per second) or continuously. A straightforward approach: Each tick, for each unit: 1. If the unit is alive: - If it’s a support unit like Medic, find ally conditions for healing. - Otherwise, find if there is an enemy in attack range and line-of-sight. If yes and attack cooldown <= 0, then **Attack**: reduce enemy HP by damage, reset cooldown. Switch to attacking state (which might just mean it’s not moving this tick). - If no enemy in range, then **Move**: advance toward enemy side (for player units, increase x position to the right; for enemy units, decrease x to the left) by its movement speed * deltaTime. - Also possibly check if an enemy is directly in front (adjacent) blocking movement (like two melee engaged). If so, stop movement (they’re effectively in combat range, just attack range might be 1 tile so they can attack). 2. Decrease attack cooldown timers for next tick. 3. If HP <= 0, mark unit as dead and remove or stop updating it. - Collision: Since units in different rows don’t collide, we only consider collisions in the same row. If two opposite units enter the same tile (or their positions x overlap beyond a threshold) in one row, they are now in melee range. They should stop moving further (they’ve engaged). At that point, even if their range is 1 (melee), they can attack. If one dies, the other can resume moving. If a ranged unit gets bumped into by an enemy because it didn’t stop (rare if logic is correct, since ranged should stop before that), it effectively is in melee and might try to move away if we coded that (could do: ranged tries to maintain distance – e.g., if enemy comes within 1 tile, maybe ranged unit attempts to backpedal some tiles if possible). - The above logic yields an automated battle where units basically charge and fight, similar to many auto battlers or Clash Royale troop behavior (though our perspective is side-on).

First Battles vs AI and Future Multiplayer

AI Battles (Current): As mentioned, the initial implementation uses AI enemies. This will effectively act as a single-player campaign or practice mode. The difficulty will ramp up gradually. We can theme the AI as various factions or raiders the player encounters: - Maybe call the first opponents “Space Pirates” (easy), later “Alien Mercenaries” (medium), etc., to give some narrative context. - Have 10-20 progressively harder levels for the player to conquer, which should be enough for a proof-of-concept. On completing these, the player’s base likely will have grown significantly.

Matchmaking (Future): In the final vision, battles will be against other players of similar skill (via an online matchmaking system). We design the rating system now (Battle Score) to make this feasible. When multiplayer is implemented: - A matchmaking service will pair players with similar Battle Score or tier. They will each pick their units and formation (likely simultaneously on their own clients). - The server (using AWS GameLift or similar) would receive both players’ unit setups and simulate the battle (to prevent cheating), then send the result back to both clients. Our game logic therefore should be able to run independently of the rendering at the same outcome given the same inputs. This means avoiding any non-deterministic behavior (or controlling random seeds). - For now, we simply note that our single-player AI battles approximate this by using fixed AI layouts. We ensure the battle system is coded in a modular way: one

function takes two armies as input and processes the fight. That same function could later run on a server with two human players' data. - **No account system now:** Currently, since it's single-player, we might not implement persistent accounts or login. Progress can be saved locally (e.g., browser local storage or an in-memory state for the session). Later, adding user accounts would allow cloud-saving of base progress and enable true online play. We should keep the code organized so that plugging in a user profile system later won't require rewriting the core logic (e.g., separate game state management from rendering).

Rewards and Progression Scaling

Winning battles is the primary way to earn **Galactic Credits** and sometimes other rewards. To keep players motivated: - Each victory yields a chunk of credits that scales with the difficulty of the battle. For example, beating a very easy opponent might give 50 Credits, whereas a high-level opponent might give 500 or more. This ensures that as upgrades become more expensive, the rewards from tougher battles also increase correspondingly. - We can also introduce reward bonuses like **first-time rewards** (the first time you beat a particular AI or level, you get an extra one-time bonus like a stash of metal or a new unit unlocked). - Optionally, incorporate a **star rating** or achievement for each battle if we had a campaign map (e.g., beat a level without any unit dying = 3 stars). That could be future detail for PvE campaign mode. - **Battle Score** (Rating): Winning increases your rating by a set amount (or more if the opponent was higher rated), losing might decrease it slightly. This keeps track of your progress in the combat arena. It could be displayed as a rank (e.g., "Bronze III") to give a sense of accomplishment. In single-player context, it's mostly for matching you to appropriate AI. In future, it directly correlates to PvP ranking. We should balance it to not rise too quickly or players will hit tough opponents too soon. - **Unlocks:** Progress in battles could be tied to unlocking content. For instance, reaching a certain battle score or beating a specific AI level might unlock the next tech or unit type to research in your base. This integrates the two modes: you not only need base upgrades to fight better, you need battle success to access new base upgrades (could be via an item reward or just a design choice: "After defeating the Alien Overlord boss, you recovered a prototype plasma weapon – you can now research Plasma Mages!"). This provides a narrative justification for progression and ensures the player experiences both modes, rather than just farming one mode endlessly.

Art and Audio Details

This section provides additional guidance on the aesthetic components of the game, ensuring a cohesive cartoon space theme across visuals and sound. While actual asset creation is separate from code, describing them here will help any AI generation or artist to produce the right style.

Visual Style & Theme

Overall Style: Cartoonish, colorful, and slightly exaggerated. Think along the lines of popular mobile base-building games (Clash of Clans, Boom Beach) but with a sci-fi twist, or even Saturday-morning sci-fi cartoons. Characters have a friendly, approachable look (even the enemies are not too grotesque – they can be aliens but in a fun way). The world is vibrant – space doesn't have to be all black; we use purples, blues, and other hues for nebulas, etc.

Home Base Visuals: The home base background should depict an alien planet environment. For example: - **Base Background:** A panoramic view on the planet's surface. One concept: The base is on a red-orange desert plateau with strange rock formations in the distance. The sky is a gradient from purple to black with stars visible. A ringed planet or large moon hangs in the sky to reinforce the space setting. Perhaps a couple

of alien plants or glowing crystals stick out of the ground near the edges. The base structures will sit on this terrain. - There might be a landing pad or ground textures under the buildings (metallic floor pieces around the HQ, connecting paths between structures drawn in a simple way so it doesn't look too bare). - We could incorporate small animated details: e.g., a rover vehicle occasionally driving by, or a spaceship flying past in the sky periodically. These give life but are not interactive. - The lighting is bright and clear (like daytime on this alien world, or maybe perpetual twilight if we want stars visible – but in cartoon style, we can take liberty that stars are visible even in a stylized daytime sky). - Buildings as described before should each have distinct shapes and colors, with slight animations (e.g., the Crystal Synthesizer's crystals might pulse slowly, the Radar dish on HQ rotates).

Battlefield Visuals: - **Battleground Background:** Possibly the same planet but a different location, or a generic landscape. To add variety, it could be a different biome: if base is desert-like, maybe the battlefield is a rocky canyon or a crater field. Alternatively, since battles could be considered training or simulated, it might even be near the base (but then background would show the base? That could be odd). Better to have a separate setting: e.g., a wide open alien plain with mountains at both sides, and the sky showing perhaps a sunset or an alien aurora. - The battlefield ground could be a flat ground with subtle grid markings for lanes: maybe slight differences in terrain for each row (like row 1 has a small ridge, row 2 a crack in ground, etc., nothing that affects gameplay but visual variety). - If possible, have mirrored small base gates at far left and far right to indicate where units come from. For example, the player's side (far left edge of screen) could show part of a bunker or gate from which your units step forward. The enemy side (far right) could show an enemy bunker or just wilderness if they are spawning from off-screen. - Keep the cartoon style: e.g., if there are craters, they have bold outlines; if there are alien plants in the background, they look whimsical (like a big cactus with eyeballs, purely decorative humor). - No obstacles on the battlefield itself (the field is open for units to move). Later, maybe environmental features could be added (cover, hazards), but not in first version.

Character and Unit Art: Already described per unit above. To reiterate general style: units have slightly larger heads or arms (exaggeration) to be expressive, and use bright colors for different teams (player units might have a consistent color scheme like blue accents, while enemy might have red accents, to easily differentiate). Animations should be somewhat playful – not too gory or realistic. When a unit dies, they could disappear in a funny puff of smoke or a spark, rather than graphic violence. This keeps the tone light.

Enemy Design: We can mirror the player's units for AI or give them unique skins: - For instance, instead of "Space Marine", the enemy might use "Alien Grunt" – similar role and stats but drawn as a little green alien with a blaster. Instead of a "Defender with shield", the enemy might have a "Alien Brute" with thick hide or armor. Mechanically same role. This way we add visual variety and narrative (the player is presumably human faction vs others). - Boss-like enemies (if any) could be larger sprites but we won't focus on bosses now.

Buildings Art Recap: Summarizing each building's look for clarity: - *HQ:* Futuristic dome/command tower with antennae. Perhaps blue-gray with some glowing lights. At higher levels, add extra domes or satellite dishes. Could have a small flag or emblem to personalize (maybe a star emblem for the player's faction). - *Mine:* An open pit mine or cave entrance with a drill. Brown/gray with metallic drill bits. Animation: drill head rotating, occasional sparks. - *Crystal Synthesizer:* Silvery structure with crystal clusters. Crystals maybe cyan-colored. They slowly rotate or glow. - *Barracks:* A squat building with target range, maybe a hologram soldier on top. Green or camo patterns to indicate military, but with sci-fi touches like a solar panel or comms antenna. - *Tech Lab:* White or light-colored facility with a telescope dome or a Tesla coil. Maybe a

holographic atom or planet above it rotating to show it's research-focused. - *Academy:* If present, maybe looks like a library crossed with a Jedi temple, but in cartoon style – could skip since tech lab covers research. - *Storage:* Metal storage looks like cylindrical tanks or cargo containers. Crystal storage looks like vault with glowing containers. - *Decorations (if any):* We can ignore for now, but later, cosmetic items could decorate base (flags, statues, etc., often idle games have that). - Each building should have an icon too (for menus), using a simplified cartoon icon that matches its shape.

Audio Design

While this is a written guide, describing the intended audio helps in implementation:

Music: - **Base Mode Music:** A loop of calm, spacey background music. It should be soothing but with a sense of progression/technology. Possibly a light ambient track with gentle electronic beeps or a slow melody that feels like building/management. Think of music that doesn't distract, since base mode involves reading and planning. Could have subtle sci-fi instrumentals (synth pads, light piano or theremin for the "space" vibe). It should evoke a feeling of curiosity and relaxation. - **Battle Mode Music:** A more energetic track that ramps up excitement. Possibly a faster tempo orchestral or electronic piece. For example, a driving rhythm with heroic undertones as your units clash. It should be loopable as battles might last up to a minute or two. Maybe include some "space battle" flavor like techno beats or even electric guitar riffs in a fun way (depending on style). The key is to make the player feel pumped watching the fight. - We could have multiple tracks (one per battle stage or increasing intensity), but one good loop might suffice initially. It might also be nice to have a short **Victory fanfare** and **Defeat tune** to play on result (like a triumphant chord if you win, a downbeat sound if you lose).

Sound Effects: - **Interface SFX:** Button clicks (a soft chirp or beep when menus are pressed), construction sound (when building starts, e.g. a hammering or futuristic welding sound), upgrade complete sound (a pleasant ding or fanfare to reward the player for completing an upgrade), resource collected sound (coins clinking for credits, mining sound for metal, a crystalline chime for crystals). - **Unit & Battle SFX:** Each unit type should have distinct sounds: - Marine's laser rifle: "pew pew" laser shots. - Defender melee: a hefty "whack" or energy bash for hitting with shield/sword. - Sniper: a loud "zap" or railgun crack sound to emphasize power. - Mage: magical whoosh or explosion for spells. - Medic: a soft hum or sparkle sound when healing. - Robot: clanking metal punch or servo noises when moving. - Enemy variations: alien guns might sound a bit different (maybe a more alien-like zap), aliens might grunt or screech when attacking or dying (cartoonishly). - **Explosions:** If any AoE like grenades or mage fireballs, have a satisfying boom (not too scary, but impactful). - **Unit Reactions:** Small death sounds (cartoony "ugh!" for humans, robot shutting down beep for bots, alien squeal for aliens). Possibly some battle cries or one-liners if we want personality (though that can become repetitive; we might stick to non-verbal sounds). - **Background SFX:** Could include ambient noises in base mode (wind on the planet, a distant spaceship engine, electronic hum from buildings). During battle, maybe crowd noise if it's an arena or just environmental (wind, distant thunder) for atmosphere.

The audio should not be overwhelming or too realistic; it should match the lighthearted, strategic feel of the game.

Future Expansion and Technical Notes

While this guide focuses on a single-player experience, it has been written to ensure the game's architecture will support future expansions such as online multiplayer, additional content, and monetization: - **Online Multiplayer Integration:** The game's battle logic is designed to work with two sets of units input. For future use with AWS GameLift or a similar service, ensure a clear separation between the **client** (which handles rendering and player input like placing units) and the **server** (which will handle the authoritative simulation of the battle in multiplayer). In the current single-player mode, the battle runs locally (with AI for opponent), but we can modularize it so that in multiplayer, the same code runs on a server instance. The matchmaking rating (Battle Score) and unit data could be sent to the server to match players. Essentially, the step of "choose units and positions" will remain on the client, but then those choices are sent to server to compute result. Designing now with that in mind (no client-side only shortcuts in logic) will make integration seamless. - **User Accounts and Persistence:** Right now, player progress (base state, units, etc.) can be kept in memory or local storage. For a deployed web game, using an online database would come when accounts are introduced. The code should be written to easily swap a local storage for a cloud save. Until then, perhaps implement a simple local save so a player can refresh the page and not lose progress (so testing the idle mechanics and progression is possible). This could be via browser localStorage or an export/import. - **Monetization placeholders:** Though the first iteration is completely free with no in-app purchases, we have left room for a premium currency and time-skipping mechanics. When premium features are added, the game should be ready to handle dual currencies (like Gems vs Credits) and special actions like "speed up build" (which would consume premium currency to finish a timer instantly). We have not deeply integrated these yet (to avoid complicating the initial coding), but building the foundation with time-based mechanics means adding a "speed up" is straightforward (just reducing a timer if currency paid). - **Scalability of Content:** The design encourages adding more unit types, building levels, and battle content. The code should therefore be data-driven where possible (e.g., define units in JSON or config so new ones can be added without rewriting logic, same for buildings). That way, the developers or AI can expand the game easily by adding to the data tables. - **AI Improvements:** Currently, the AI for enemy simply mirrors player mechanics with possibly some predetermined formation. In the future, AI could be made smarter or have varied personalities (aggressive rushers vs turtling healers, etc.). For now, a basic AI that picks a decent formation (perhaps similar to how a real player might) is sufficient. We can script a few enemy templates rather than a truly adaptive AI to keep it simple and predictable for the player.

Conclusion

Space Base Showdown offers a compelling blend of strategic base management and automated combat. With its approachable cartoon sci-fi style, players of all ages can enjoy building a space base, training a quirky army of units, and watching their strategies play out in exciting battles. This design document has detailed every aspect: from the layout of the base and functionality of each building, through the stats and behaviors of every unit type, to the flow of battle and the integration of systems that reward the player and allow progression. By following this guide, a developer or generative agent should be able to implement the game step-by-step, ensuring that by the time it's done, a player can launch into the game, intuitively build up their base, and immediately jump into battles. The first iteration focuses on single-player content against AI enemies, providing a controlled environment to fine-tune mechanics and balance. It sets the stage for future updates like multiplayer matchmaking, more units, and competitive features.

With this thorough plan, **Space Base Showdown** is ready to be brought to life. Players will soon be able to construct their cosmic colonies, assemble their dream team of space troops, and engage in epic

showdowns among the stars – all without needing to manually control the battle, because the satisfaction comes from clever preparation and seeing the plan unfold. Good luck on the coding journey, and have fun creating this space strategy adventure! [9] [3]

---

[1] Hustle Castle: Medieval games - Apps on Google Play
https://play.google.com/store/apps/details?id=com.my.hc.rpg.kingdom.simulator&hl=en_US

[2] [7] Battle Legion - How to Launch and Grow a Strategy Game - Deconstructor of Fun
https://www.deconstructoroffun.com/blog/2020/8/4/battle-legion

[3] [4] [5] [6] [8] [9] Battle Legion: Mass Troops RPG Beginner's Guide - Learn the Basics of Mass Troop Battles | BlueStacks
https://www.bluestacks.com/blog/game-guides/battle-legion/blmt-beginners-guide-en.html