Omer Blau - 209364280

Part One: Game Design

Choose a game for the assignment

(The selected game will be used for the following exercises)
The game I chose is ***Flash Element***, a Tower Defense type game.

Define the components in the game.

Each component should include a short description (1–2 sentences, no more).
You must also specify the preferred storage type for the component: **Sparse**, **Packed**, or **Tag** (i.e., no storage).

- **Spares**
  - Position – x,y grid coords
  - Target – current enemy being aimed at
  - GoldBounty – gold awarded when killed
  - UpgradeLevel – tower tier (0-3)
  - Renderable – sprite/animation handle
  - InterestBonus – extra interest % for gold bank
  - BonusVsImmune – multiplier applied to damage vs immune creeps

- **Packed**
  - Velocity – movement vector per tick
  - Speed – scalar tiles / sec that a creep travels
  - Health – current HP value
  - Damage – hit damage dealt on contact
  - Range – attack radius in tiles
  - PathProgress – index along waypoint list
  - SlowEffect – slow % and remaining time
  - SplashRadius – area-of-effect radius
  - Lifespan – seconds until auto-despawn

- **Tag**
  - Projectile – marks projectile entities
  - Tower – marks tower entities
  - Creep – marks enemy creeps
  - Immune – cannot be slowed (for creeps)
  - AirUnit – flying unit flag (for creeps)
  - CanHitAir – marks towers with the ability of hitting also air units (assuming ground is the default)
  - AirOnly – for a single tower which can hit only air creeps
  - PlacingTower - to make a tower ghostlike and have it follow the mouse and ready to place

Define the entities in the game.

Each entity should include a short description (1–2 sentences, no more), along with a list of components it contains. Note: Components can be **added or removed from an entity dynamically**. You should mark such components as dynamic (temporary, optional, event-based, etc.).

I listed the shared components for towers and creeps once to avoid repeating them. It's just for readability—I'm still treating each as a separate ECS entity. When creating them, I'd add the common components first, then add the unique ones depending on the type.

- **TOWERS**
  - **All Towers - (**to not repeat these many times in the doc)
    - Position
    - Renderable
    - Tower
    - Range
    - Damage
    - Target (dynamic)
    - UpgradeLevel
  - **ArrowTower** – cheap single-target ground/air tower
    - CanHitAir (dynamic)
  - **CannonTower** – area-of-effect ground tower
    - SplashRadius (dynamic)
  - **AirTower** – specialized tower that targets only flying enemies
    - AirOnly (dynamic)
  - **WaterTower** – fast tower that slows creeps on hit
    - CanHitAir (dynamic)
  - **FireTower** – AOE tower with bonus vs immune creeps
    - SplashRadius (dynamic)
    - CanHitAir (dynamic)
    - BonusVsImmune (dynamic)
  - **EarthTower** – extremely high single-target damage with slow fire rate
  - **RocketTower –** long-range, high-damage tower for endgame
    - CanHitAir (dynamic)

CREEPS

- **All Creeps**
  - Position
  - Renderable
  - Creep
  - Velocity
  - PathProgress
  - Health
  - GoldBounty
  - SlowEffect (dynamic)
  - Immune (dynamic)
  - Speed
- **GroundCreep – enemy that walks the path (boss fall in here too)**
- **AirCreep – flying enemy immune to some towers**
  - AirUnit (dynamic)

- **PROJECTILE – entity representing a shot fired by a tower**
    - Position
    - Velocity
    - Damage
    - Lifespan
    - Projectile
    - Renderable
    - SplashRadius (optional)

- **GAMESTATE – singleton holding global player stats and upgrades**
    - InterestBonus

Define the systems in the game.

Each system should include a short description (1–2 sentences, no more), and a list of components the system operates on.
Note: A system may have **optional components**. You should specify the **required component combination** (the system will only operate on entities containing all of these), and also describe **optional behaviors** when additional components are present.
For clarity, each section must be detailed separately (do not combine them), and each entity/system should be listed **separately** after the general list of components.

It is sufficient to describe the **general mechanics** of the game; there's no need to go into every detail or every option (e.g., for *Scorched Earth*, there's no need to describe each tank and weapon, just the possibility of defining different types).

- **SpawnSystem (event-triggered)** – spawns new creep entities at the start of each wave
    - (no required components – triggered by a button and generates creeps by the “levels table”)

- **PathNavigationSystem** – updates creep velocity toward their next waypoint
    - Position
    - Speed
    - PathProgress
    - Creep
    - SlowEffect (optional)

- **MovementSystem** – moves entities based on their velocity
    - Position
    - Velocity

- **TargetingSystem** – towers find and lock onto valid creeps in range
    - Position
    - Range
    - Tower
    - CanHitAir (optional)
    - AirOnly (optional)

- **ShootingSystem** – towers with a target spawn a matching projectile
    - Position
    - Damage
    - Tower
    - Target
    - SplashRadius (optional)
    - BonusVsImmune (optional)

- **ProjectileCollisionSystem** – checks projectiles for collisions and applies hits
  - Position
  - Damage
  - Projectile
  - SplashRadius (optional)
- **DamageSystem** – reduces health when hit and handles creep deaths
  - Health
  - GoldBounty (optional)
  - Immune (optional)
- **SlowEffectSystem** – updates and removes expired slow effects
  - SlowEffect
- **UpgradeSystem (event-triggered)** – upgrades tower stats when triggered by the player
  - UpgradeLevel
  - Tower
  - (SplashRadius, BonusVsImmune, etc. optional – applied based on tower)
- **InterestSystem (event-triggered)** – adds interest gold at the end of each wave
  - InterestBonus
- **CleanupSystem** – removes projectiles and other entities when expired
  - Lifespan
- **RenderSystem** – renders all visible entities to screen
  - Position
  - Renderable

Part Two: Infrastructure Tests

Create a test file named **`tests.cpp`** (similar to the existing `tests.cpp` file available on GitHub) with **two new test functions**:

1. A test for **DynamicBag**.
2. A test for **PackedStorage**.

In each function, perform **2–3 tests** of the relevant class and ensure that the code runs and the tests pass successfully.

For the purpose of testing, you may **assume that all fields in these classes are accessible** (you may change `private` to `public`).

While I've attached the test.cpp file to the submission, for convenience, I've also uploaded it to my Google Drive: Download test.cpp.

I wrote this document in English for clarity and efficiency, both during research and implementation. Writing directly in English helped avoid unnecessary translation between Hebrew and English, especially since the project itself (and any future publication to GitHub) would likely be in English anyway 🙂

I spent a lot of time researching, playing and understanding the mechanics in this game in order to make the upcoming implementation smoother and more organized. If you're interested in diving deeper into the background, feel free to check out the research notes I compiled here: Flash Element TD Research