Name: Yoav Eliav
Id: 312498207

Aeneid

Architecture

1. project Background

Aeneid is a 2d game which tries to return to the nostalgic aspect of the roleplaying game genre.

Today, most of the roleplaying games that are produced are 3d games, a games with a lot of features which makes the game lose its simplicity and gameplay experience.
Mainly because of this reason I decided to create Aeneid, I want to create a game for fans of nostalgic games.

The story of Aeneid takes place in a fictional world, a world where there are a lot of other species and planets that the player will have to explore during the game.

2. Project Scope

what will done in scope:

- saving/loading the game state.
- a character controller that acts differently based on gameState and player input.
- a spaceship controller that acts differently based on gameState and player input.
- AI enemies with different behavior for each enemy Type.
- a trading system that allows the player to buy and sell things.
- a different world map for each planet.
- Cross-platform support: macOS/Windows
- player inventory.
- trading system

what won't be done:

- multiplayer option
- weapons mechanic
- Player Customization
- Day and Night Cycle
- changing the game performance setting like resolution, frame rate, etc...
- control the game difficulty.

what may be done:

- level point mechanism: the player will be able to gain level points at each level and choose how to spend them.
- one on one game mode: the option to fight one vs 1 against a known enemy.
- combo system.
- cloud save/load.
- joyStick support.
- ability to buy a new spaceship with different stats.

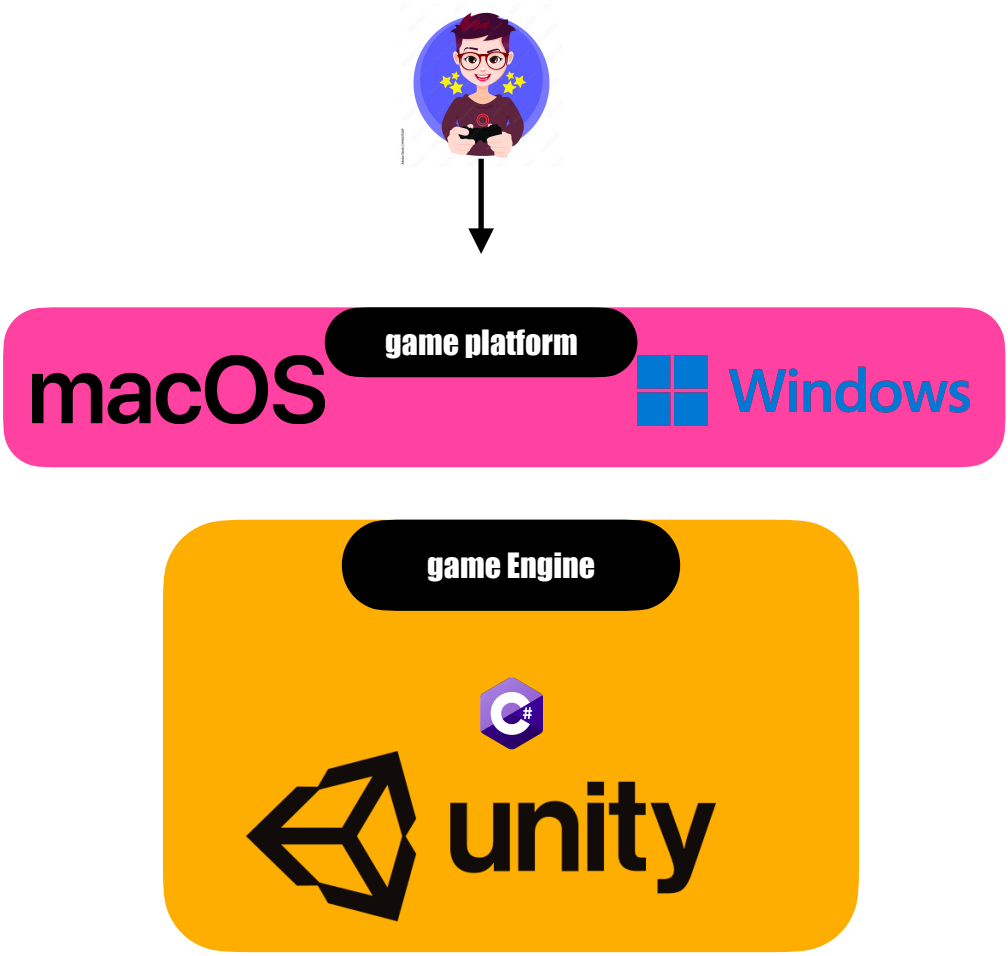
3. High-Level Requirements (Design)

- the player of the game should be able to control the game character and the spaceship.
- each map (planet) of the game should include monsters/npc.
- each Enemy type at the game should have different combat AI
- the player should be able to manage his skills and items he got.

4. Who are the users?

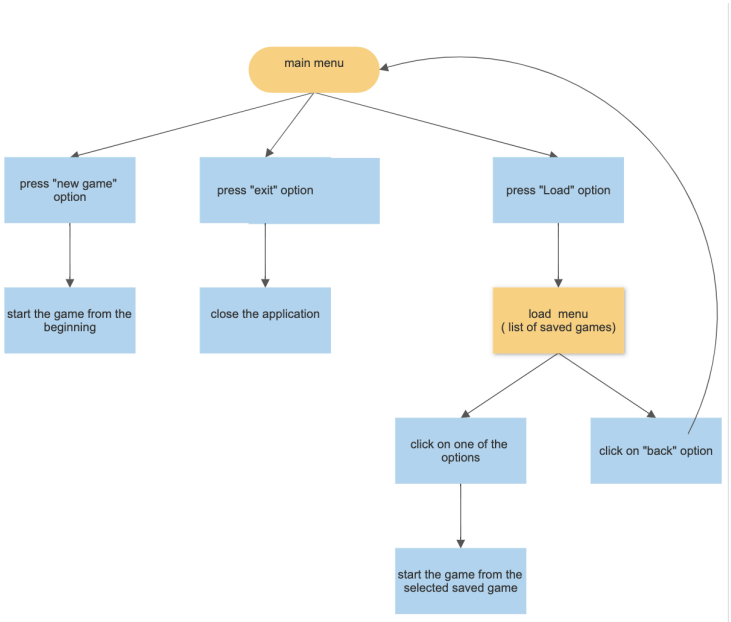
there is only one type of user: the player himself who plays the game.

5. Project entities (Architecture)

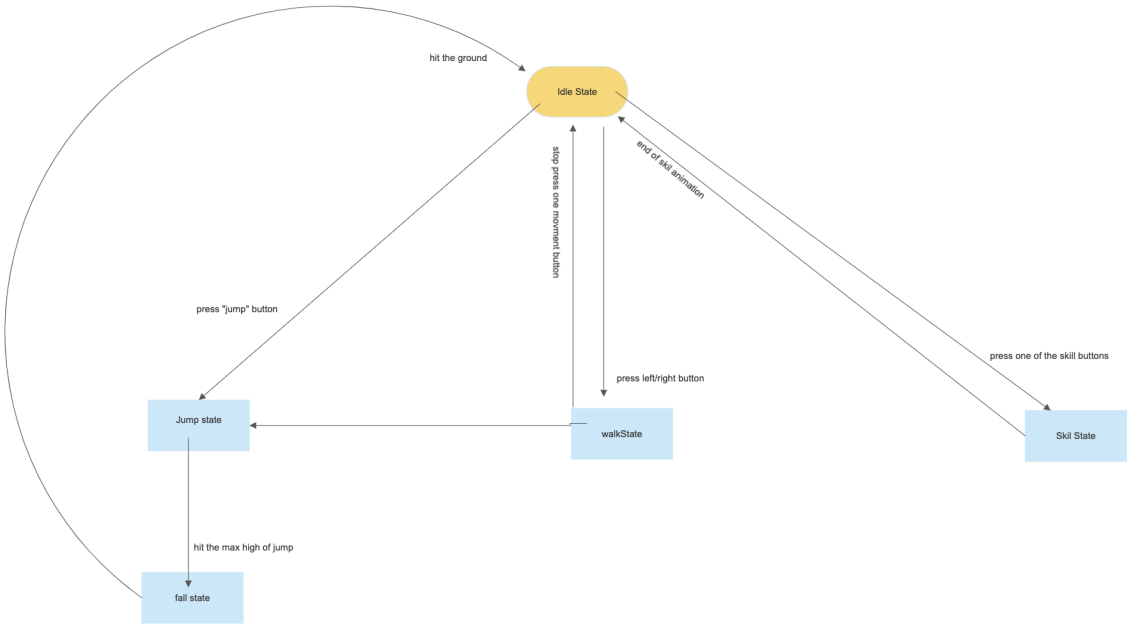


6. Typical user flows

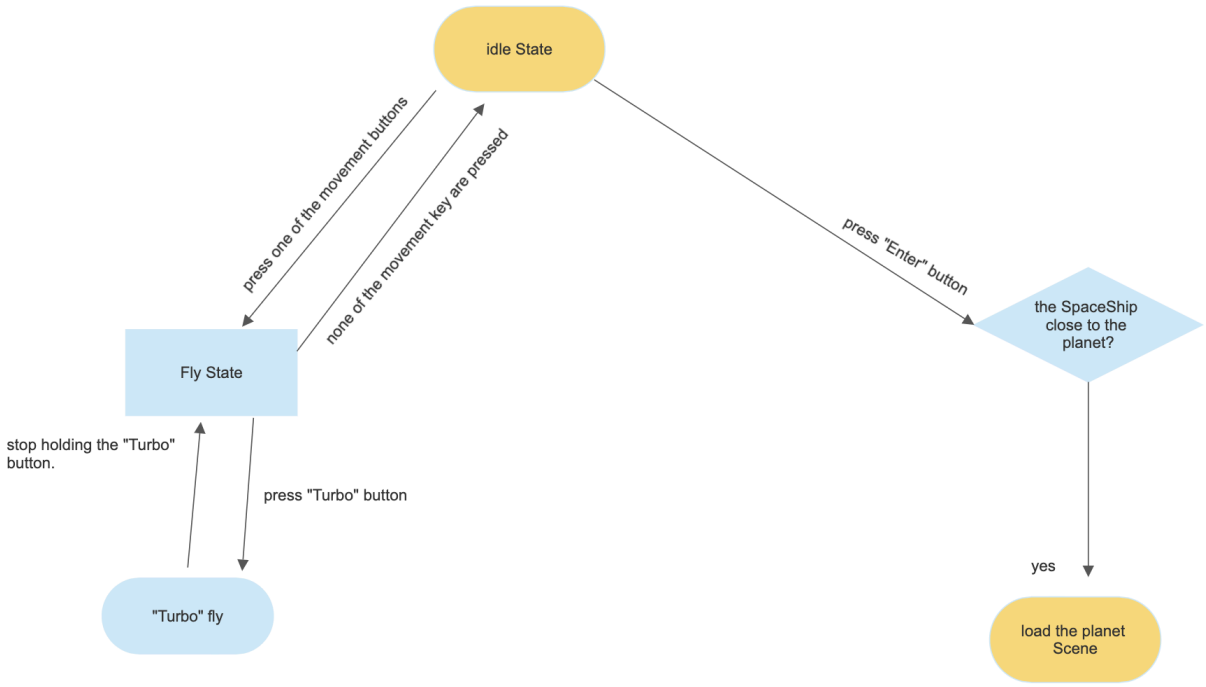
Main Menu flow



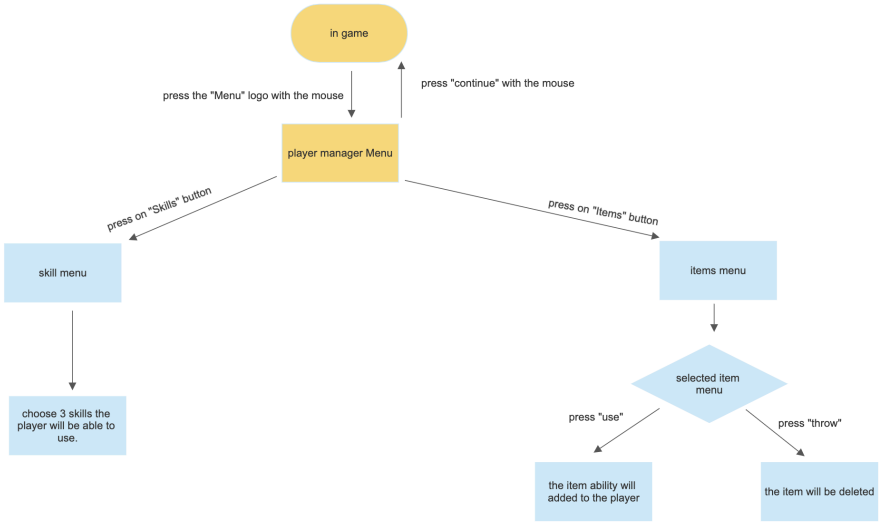
Player movement flow



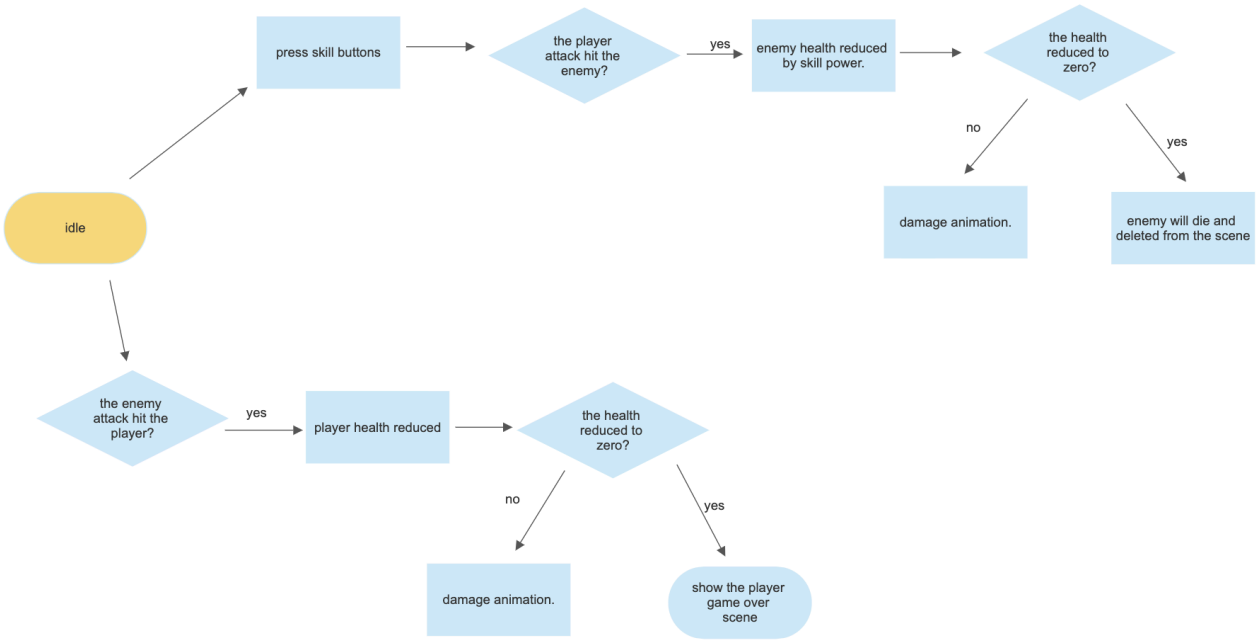
spaceship control flow



player inventory Menu



Player Combat flow



7. Epics and user stories and test plans scenarios

Epic 1: player movement

user stories:

- As a player I want to be able to walk with my character.
testPlan: check if the player reacts to movement inputs, also check the player reaction when in different states like jumping or at doing skill.

- As a player I want to be able to jump
test plan: check if the player reacts to the jumping button, also check how the player reacts when not touching the ground.

- As a player I want to be able to do one of 3 available skills
test plan: the test will include the following things:
* check if the player is doing the correct skill he chooses.
* check that the player can do skill only when idle
* check that the player can do the same skill only after some cooldown.

- As a player I want to be able to interact with npc around the map.
test plan: i will check the following scenarios:
* check that the player is unable to interact with the npc if he isn't close to them.
* check that the player interact option appears only when close to the npc.

- As a player I want that my player will fall when not touching the ground (fall state)
test plan: create a gizmo that will show when the player does not touch the ground, check if when the gizmo appear red(meaning not touching) the player enters a fall state.

Epic 2: player UI

- As a player I want to be able to save my game.
test plan: check if when saving the game the saved file is added. also check with unity inspector that the game as the same stats

- As a player I want to be able to choose which saved game i want to continue:
test plan: check sample of different saved games and check with unity inspector that all the stats belong to the save file.

- As a player I want to be able to be able to change my 3 main skills
test plan: i will test in 2 ways:
* use unity inspector to see if the skill is changing.
* check during the game if the skills change.
i also check that the player will be unable to change the skill he didn't unlock

- As a player I want to be able to see my stats like health bar and experience points.
test plan: the test will be in 2 methods:
1) using unity inspector to change the health bar and experience bar and see if the changes apply during the game.
2) check during game events if the health bar and experience bar update.

- As a player I want to be able to manage my items in the backpack, use them or throw them.
test plan: delete the items and check with the unity inspector if the item deleted from the memory of the game, if the player uses the item check if the effect of the item has applied to the character.

Epic 3: Enemy Ai

- As a player I want that when the enemy sees me he will chase me until I am out of his sight.
test plan: to test this i will use unity gizmos and check in scene mode to we when the enemy chase after the player and when note

- As a player I want that when the enemy sees an opportunity to attack me he will attack me.
test plan: to test this i will use gizmo to see if the enemy sees opportunity to attack, if so check if the enemy try to attack.

- As a player I want the enemy will try to dodge my attacks
test plan: the test will be in two ways: first, check using a loop that invokes the enemy get hit method to check if the enemy tries to dodge, Second, create a battle against the enemy to see if he tries the dodge during the battle.

- As a player I want that the enemy will patrol around the map
test plan: using unity scene mode to watch if the enemy patrols during game.

- As a player I want the enemy to have health bar which indicate the enemy's life.
test plan: the test will be the same as the test of player health bar.

***note about Gizmos and unity inspector:**
- Gizmos are used to give visual debugging or setup aids in the Scene view.
- Unity inspector is a window that lets the creator of the game, view and edit properties and settings for almost everything.