IE201 Project Part 3 Report

Group 6

08.01.2023

Tevfik Buğra Türker - 2019402120

Yusufcan Özkan - 2019405105

Oğuzhan Engin - 2019402009

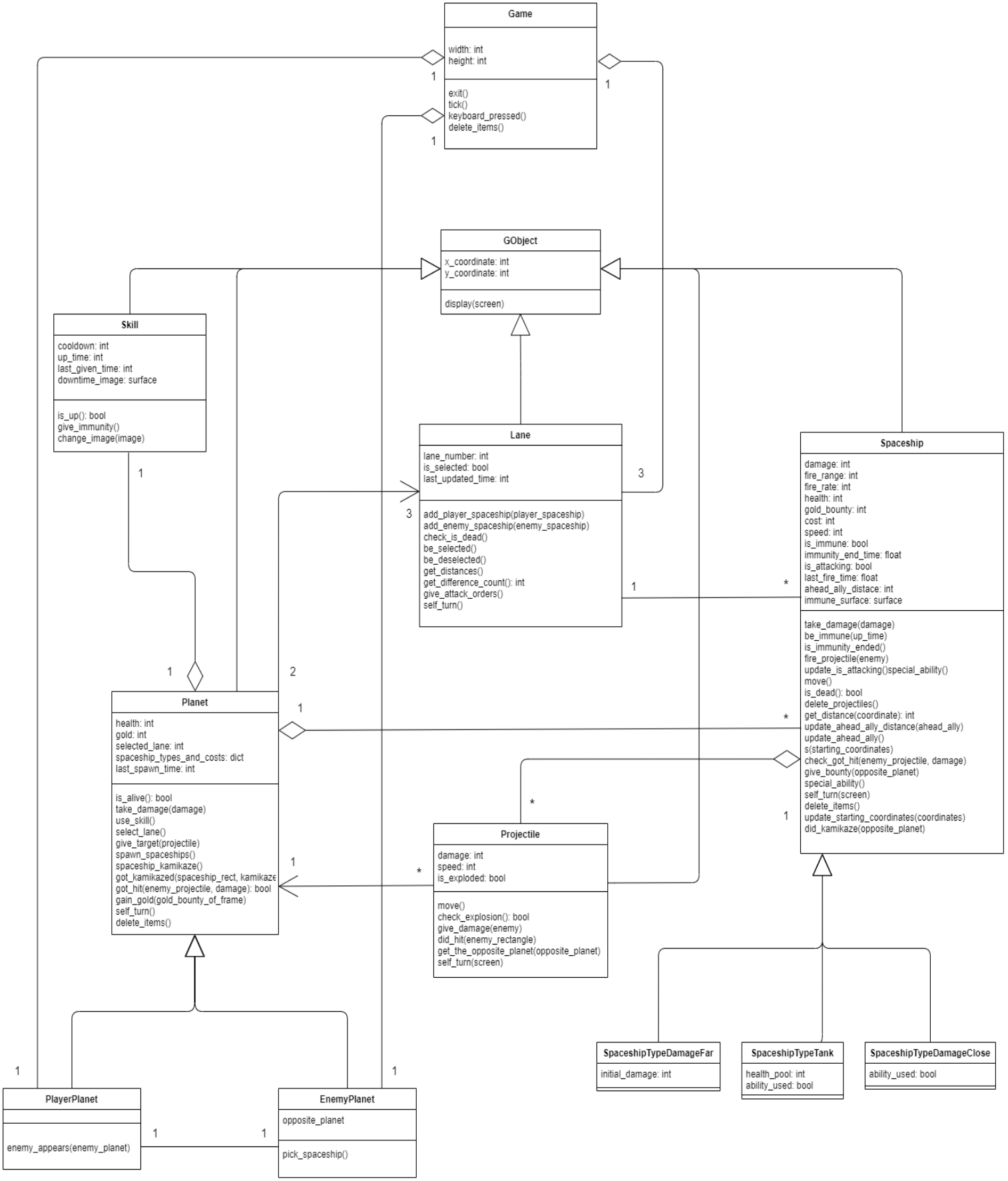
Hasan Yağız Kılıç - 2019402147

CONTENTS

Description of the Game

In the game, there are two enemy planets. These planets can create different types of spaceships, with certain advantages and disadvantages. These spaceships come across on a battlefield of three straight lanes. Spaceships created in a lane fight against spaceships coming from the same lane. Planets are located on both sides of this area and must be protected from the enemy throughout the game. The goal is to reach and destroy the enemy planet by eliminating the enemy spaceships first. In order to do this, the player should choose the right type of spaceships and send them to the right lane. Since the lane of a created spaceship cannot be changed later, line selection has a crucial place in winning the game. Thus, an effective strategy is needed to surpass the enemy. Besides the lane selection, the spaceship selection is just as important. Spaceships, which differ in features such as range, fire rate, health, speed, should be chosen in the most effective way, considering the conditions of each lane. With every enemy spaceship destroyed in the game, some money is earned. With the money earned, new spaceships can be purchased. In addition to all these, there is a special skill that when activated can change the fate of the game. With the usage of this skill, all friendly spaceships become immortal for a certain period of time. This skill can change the course of the game with its power. However, since the cooldown time is quite long, it is important to choose the right time to use the skill. If a planet takes too much damage from enemy spaceships and loses all of its health, it will be destroyed, and the game will be over.

Class Diagram



Explanation of the Classes

Game:

Game is the main class of our game that is responsible for making initial calls to other classes and starting chained functions that will be called each frame. This class has references to both player and enemy planets, as well as to three lanes.

GObject:

GObject is the parent class to all classes that are in our game except for the Game class. This class has coordinates and image attributes which are common for all classes except for the Game class.

Planet:

Planet is the parent class of two classes: PlayerPlanet and EnemyPlanet classes. These child classes are the two sides of the battle. Both sides own a certain amount of gold. With that gold they have the ability to spawn different types of spaceships to selected lanes. Both sides have the ability to use the skill which makes spaceships immune to projectiles throughout a time interval. Planet class has a certain amount of health. Every time an enemy spaceship reaches the planet, the amount of health decreases by the damage it takes from enemy. When the health drops to zero, the game ends.

PlayerPlanet:

PlayerPlanet is the class that the player of the game controls. If its health goes to zero, the game ends and the player loses. The player can control this class in order to select lanes, spawn spaceships and use the skill.

EnemyPlanet:

EnemyPlanet is the class that is controlled by an algorithm that decides the enemy's actions such as choosing a spaceship and the lane in which it will be spawned and when to activate the skill. If the health of the enemy planet goes to zero, the game ends and the player wins.

Skill:

Skill grants immunity to projectiles for all friendly spaceships for a limited amount of time. However, the skill can’t be used successively due to the cooldown time, which is the minimum amount of time that should pass in order to activate the skill again. Therefore, the skill doesn't have a full uptime.

Lane:

Lanes are the paths that spaceships can move through in the game. There exist 3 lanes in the game.

Spaceship:

Spaceship is the parent class for 3 different types of spaceships: SpaceshipTypeDamageFar, SpaceshipTypeTank, SpaceshipTypeDamageClose. These spaceships are specialized in different areas. They differ in damage, fire range, fire rate, health, cost, bounty, and speed. Also, each one of them has a distinct special ability. When their health attributes drop to zero, thedestroyed. Moreover, they can fire projectiles which inflict damage on enemy spaceships and the enemy planet.

SpaceshipTypeDamageFar:

Type of the spaceships that are specialized in dealing damage from long range. They deal more damage when opposite spaceship is further.

SpaceshipTypeTank:

Type of the spaceships that are specialized to withstand high damage. When their health points drop to the critical threshold, they restore some of their health.

SpaceshipTypeDamageClose:

Type of the spaceships that are specialized in damage. They have high speed. When they pass forward the critical line, they gain extra damage.

Projectile:

Projectiles are fired from spaceships and damage the opposite planet or its spaceships. They move along the paths bounded by the lanes.

Relations Among the Classes

Game class has an aggregation to PlayerPlanet, EnemyPlanet and three of the Lanes. Game class controls the flow of the game with these classes.

The GObject class has an inheritance relationship with the following classes: Skill, Planet, Lane, Spaceship, Projectile. All the child classes have coordinates and image attributes which are inherited from the parent class.

Planet class has a skill. So it has an aggregation with Skill class. It also has spaceships. Spaceships are owned by a planet. Thus an aggregation between Planet class and Spaceship class is built.

To spawn spaceships, planet class should decide the lane to launch the spaceships. So the Planet class should know about the Lane class. Thus a unidirectional association is formed between the Planet and Lane classes.

EnemyPlanet and PlayerPlanet are special types of the Planet class. So inheritance relationships between Planet, EnemyPlanet and Planet, PlayerPlanet should be formed.

Spaceships have their own projectiles. Thus, an aggregation between Spaceship class and Projectile class is formed.

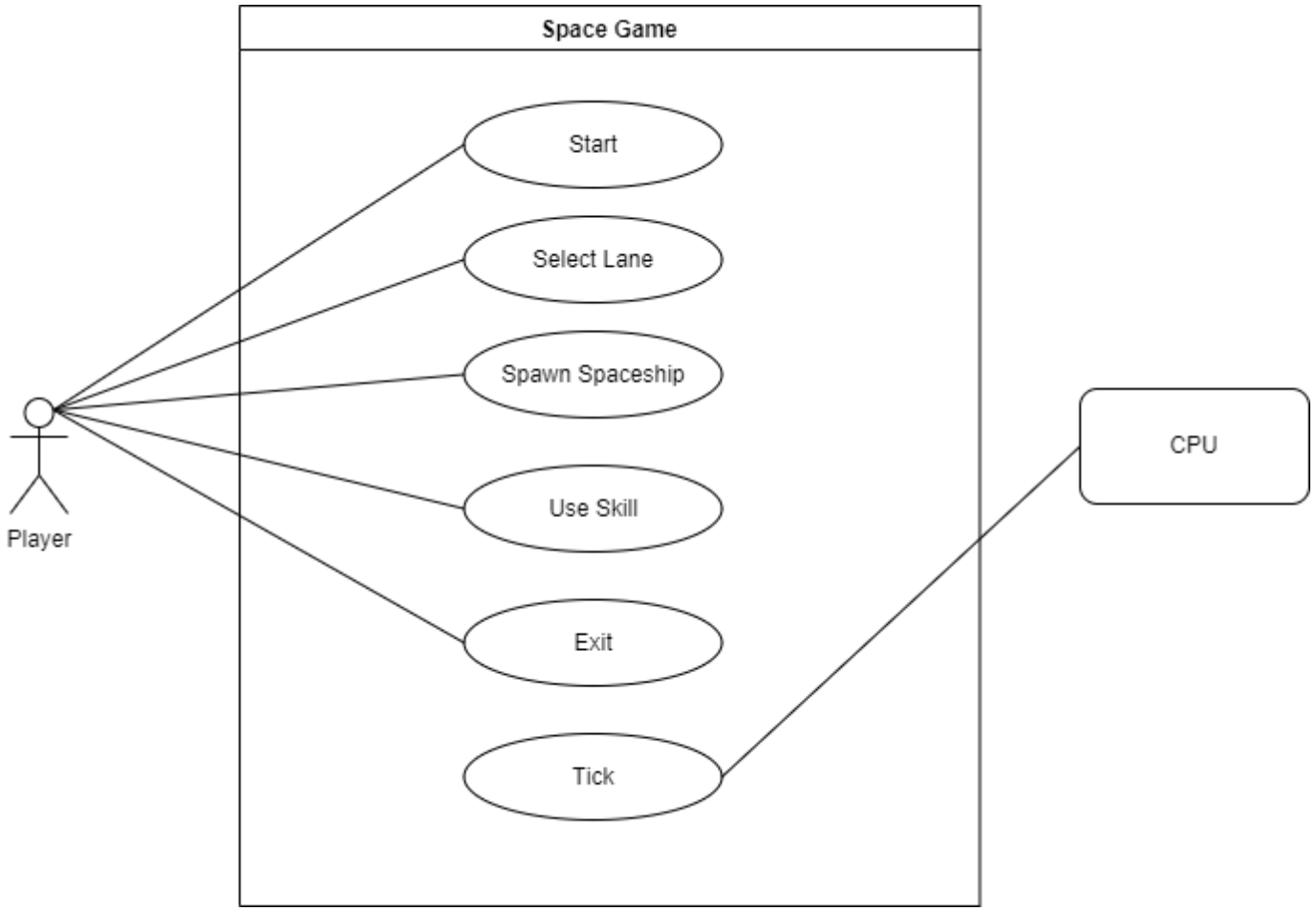
To inflict damage to both, the opposite planet and its spaceships, projectiles should know about the opposite planet. Thus, a unidirectional association is built between the Planet and Projectile classes.

Spaceship class has three special types, namely type DamageFar, Tank and DamageClose. Between the Spaceship class and these three types, an inheritance relationship should be formed.

The Lane Class holds a list of the spaceships that are currently in the lane. Thus, an association between the Lane and Spaceship classes should be formed.

In order to keep track of the golds, planets inform each other about how much gold they gained from killing each other's spaceships. Also, when Planets create new spaceships, they give opposite planet as the target. Therefore, an association between PlayerPlanet and EnemyPlanet is established.

Use-case Diagram



Explanation of the Use-cases

Start:

After the player clicks the start button, the game starts, and various game objects are created.

Select Lane:

The player clicks up or down button and chooses the lane in which the new spaceship will be spawned. After the selection, the selected lane remains selected if no other new selections are done.

Spawn Spaceship:

The player clicks a, s or d buttons and chooses the type of spaceship that will be spawned. These buttons respectively correspond to the following types: Tank, DamageClose, DamageFar. To spawn a spaceship, the player should have enough gold to purchase the selected type and at least 2 seconds should have passed after the last spawn.

Use Skill:

The player clicks space button and activates the described skill that makes spaceships immune to damage. To activate the skill, the cooldown time must be elapsed. The skill is activated only when the cooldown has passed, and the player clicks the space button.

Exit:

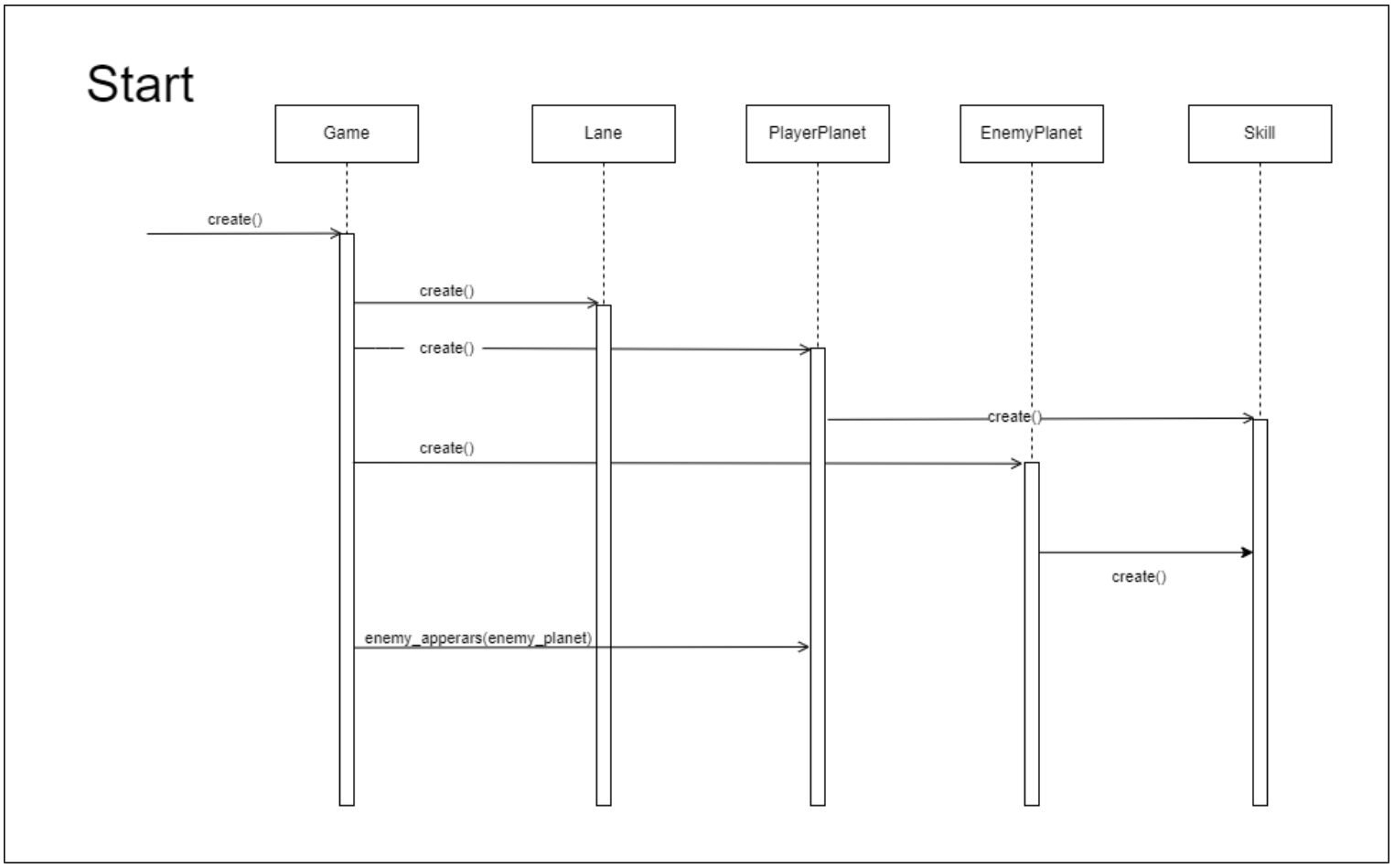
The player pushes the F button and exits the game or the game ends when the health of any planet drops to zero.

Tick:

The game is periodically updated by CPU. All the things that should automatically be done in a frame is executed by the tick including the game logic of the enemy planet which is controlled by an algorithm.

Collaboration and Sequence Diagrams

START SEQUENCE DIAGRAM



There is no function called start in our code. However, in the python programming language, the initializer function of each class basically creates that object.

1 First, the Game class is created by the init() function.

1.1 Lane class is created by the init() function.

1.2 Planet's child class "PlayerPlanet" is created by the init() function.

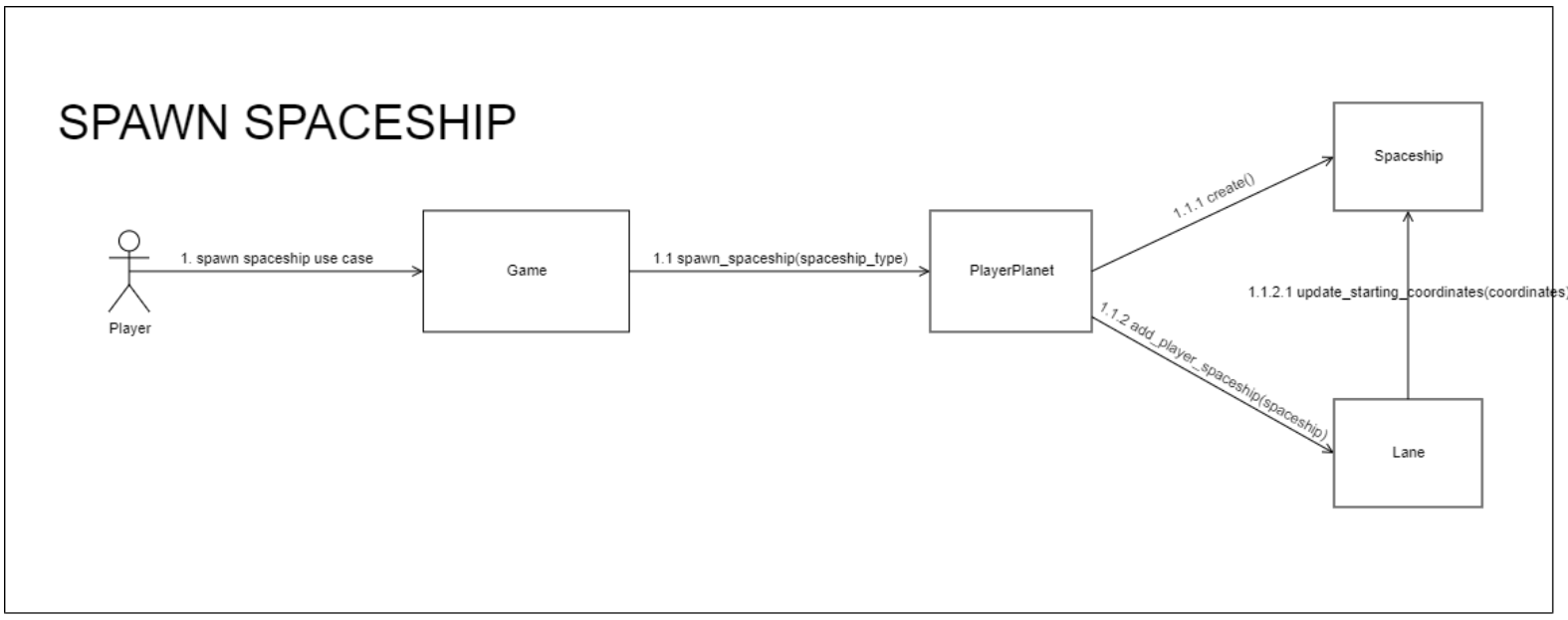
1.2.1 PlayerPlanet's skill is created by the init() function.

1.3 Planet's child class "EnemyPlanet" is created by the init () function. The reference of the PlayerPlanet is given to the EnemyPlanet since the PlayerPlanet class already exists when the EnemyPlanet class is created.

1.3.1 EnemyPlanet's skill is created by the init() function.

1.4 Both EnemyPlanet and PlayerPlanet classes need to know about each other. Since the EnemyPlanet class didn't exist when the PlayerPlanet class was created, the reference of the EnemyPlanet couldn't have been given to PlayerPlanet. Thus, the additional function enemy_appears() is called to give the reference of the EnemyPlanet to the PlayerPlanet after the EnemyPlanet object is created.

SPAWN SPACESHIP



When a player presses buttons to spawn spaceships, Spawn Spaceship use-case starts by calling the keyboard_pressed() function of the Game class.

1 The spawn spaceship use case is triggered by the player.

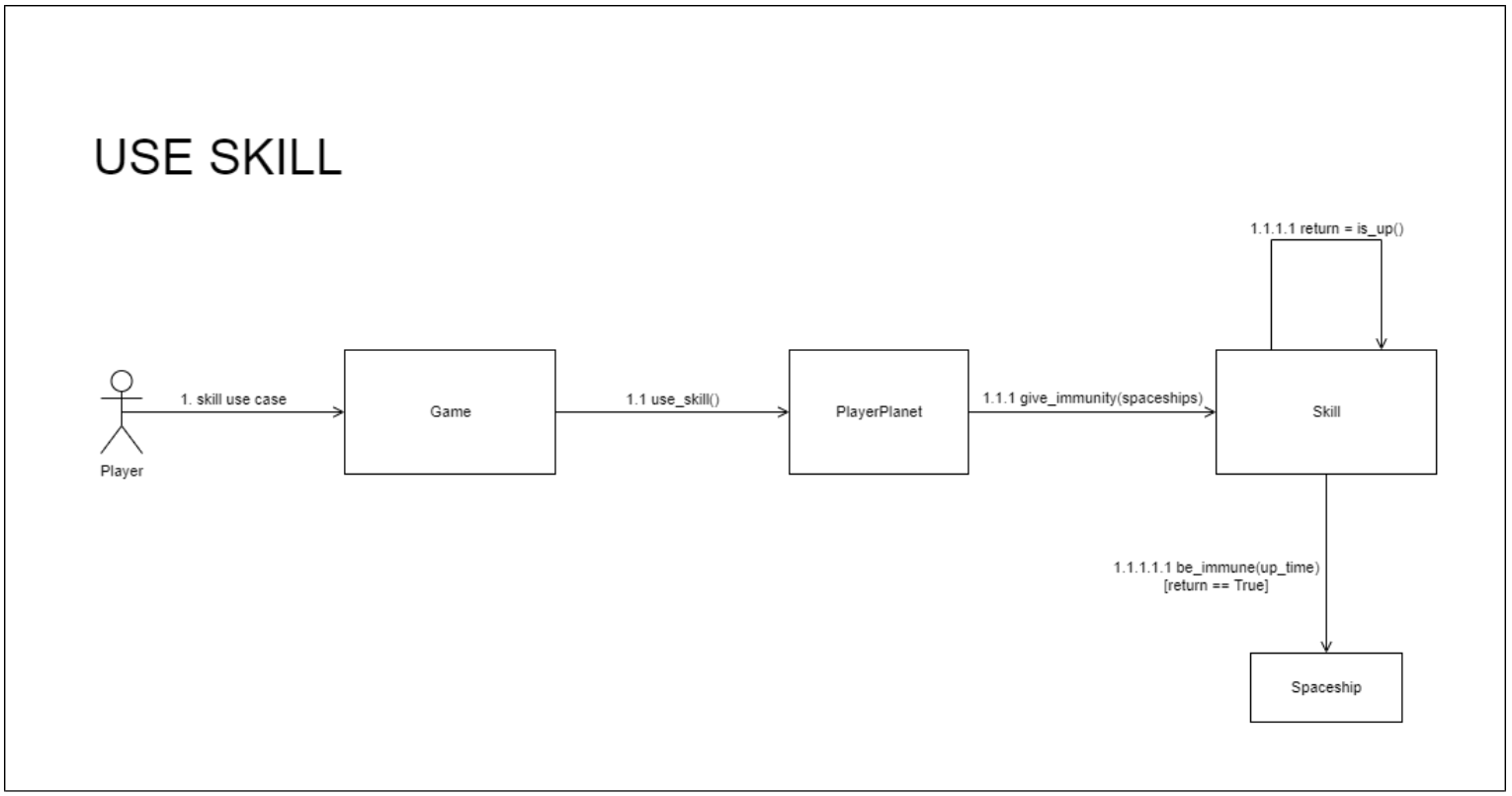
1.1 If the player presses one of the three buttons that represents each three different kinds of spaceships, the spawn_spaceship(spaceship_type) function of the PlayerPlanet object is called. Firstly, this function checks if the amount of the gold attribute of the PlayerPlanet object is not less than the cost of the chosen spaceship type. In other words, it checks if the player has enough gold to buy the spaceship. Then, the function decreases the amount of gold the PlayerPlanet has by the cost of the chosen spaceship.

1.1.1 An instance of the chosen child class of the Spaceship parent class is created.

1.1.2 From the spawn_spaceship() function of PlayerPlanet class, add_player_spaceship() function of the Lane class is called. After that point the chosen Lane knows that there is a new player spaceship on itself.

1.1.2.1 From the add_player_spaceship(return) function of the Lane class, update_starting_coordinates(starting_coordinates) function is called. Hence, the starting coordinate of the spaceship created is now known by the spaceship itself.

USE SKILL



When a player presses the skill button to activate the skill, the Use Skill use-case starts by calling the keyboard_pressed() function of the Game class.

1. The skill use case is triggered by the player.

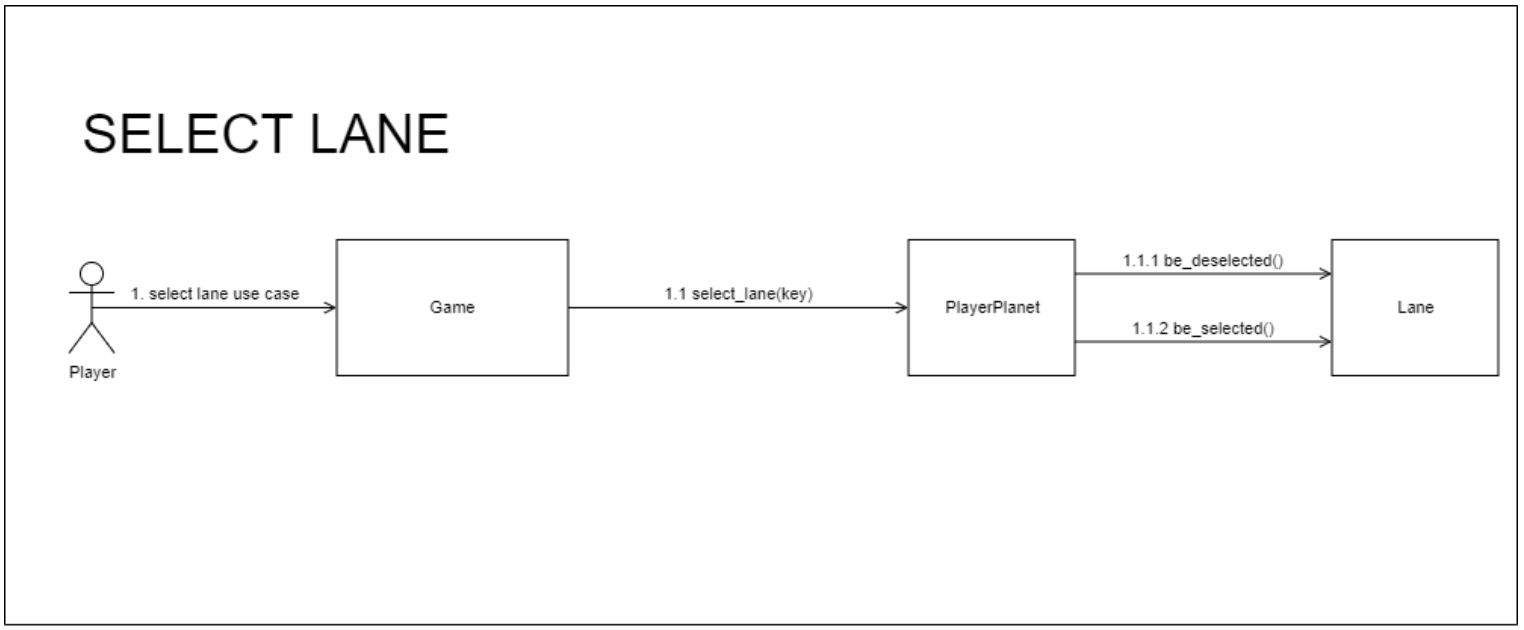
1.1 If the player presses the “space” button, which is chosen as the button that activates skill, the use_skill() function of the PlayerPlanet class is called. The use_skill() function operates and calls the give_immunity() function of the Skill class.

1.1.1 In the give_immunity() function, the is_up() function of the Skill class is called to check whether the skill is usable or not for the moment.

1.1.1.1 The is_up() function returns a Boolean value for skill being available or not.

1.1.1.1.1 If the returned Boolean value of the is_up function is True, be_immune(up_time) function makes player's all spaceships immune to damage for a certain period of time.

SELECT LANE



When a player presses buttons to select a lane to deploy spaceships, Select Lane use-case starts by calling the keyboard_pressed function() of the Game class.

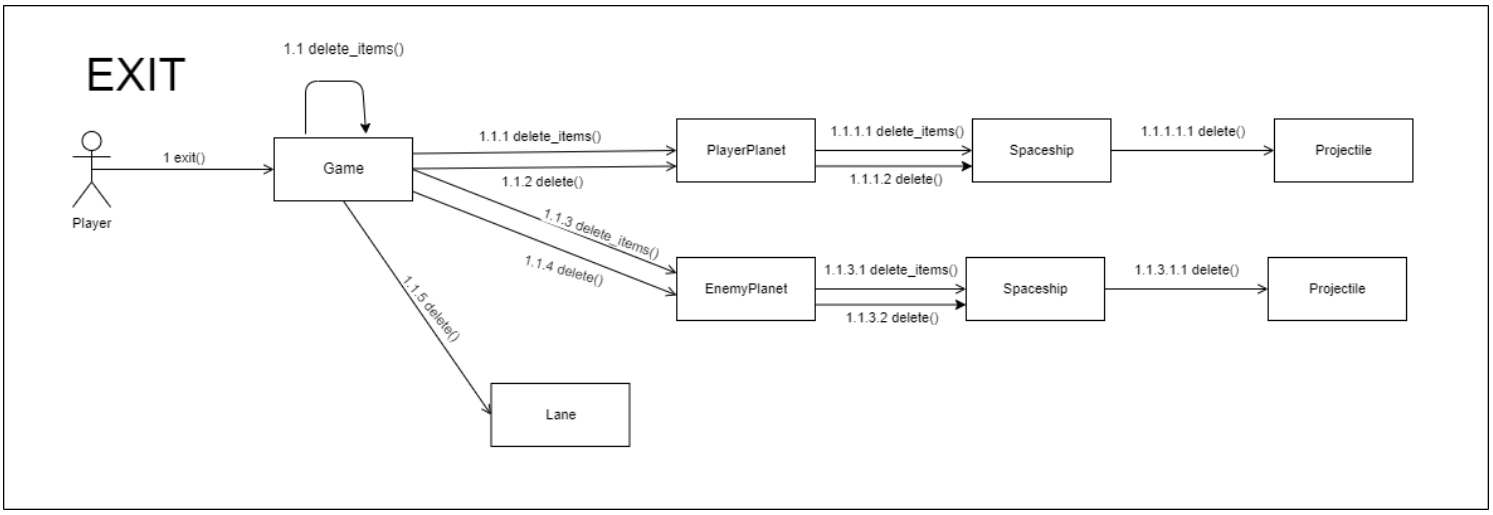
1. The select lane use case is triggered by the player.

1.1 If a player presses "lane up" or "lane down" buttons, the select_lane(key) function of the PlayerPlanet class is called and it takes the key to change the selected lane index of itself. After the select_lane(key) function is called, it calls the be_deselected() function of changed Lane and be_selected() function of the new Lane according to the selected lane index.

1.1.1 The be_deselected() function deselects the previously selected lane.

1.1.2 The be_selected() function selects the new lane for spaceships to be deployed.

EXIT



The use-case “Exit” begins with the calling of the exit() function of the Game class.

1 The exit() function of the Game class is called.

1.1 The does_exit boolean is initially False. If the player closes the game window or presses the “F” button, does_exit Boolean becomes True and the Game class calls the delete_items() function.

1.1.1 First, the delete_items() function is called in order to start deleting the PlayerPlanet class and its objects. Since we cannot relay the delete() function to other classes if the PlayerPlanet class deletes itself first, it uses delete_items() function to delete its objects first.

1.1.1.1 delete_items() function of the spaceships is called to delete all the projectiles owned by each player spaceship.

1.1.1.1.1 delete() function is called to delete the projectiles of the spaceships.

1.1.1.2 delete() function is called to delete each player spaceship after all of its projectiles are deleted.

1.1.2 After deleting all the spaceship class instances belonging to the player, the PlayerPlanet class itself is deleted with the delete() function.

1.1.3 First, the delete_items() function is called in order to start deleting the EnemyPlanet class and all the spaceship class instances owned by the enemy. Since we can't delete the rest of the classes if the EnemyPlanet class is deleted first, deleting the EnemyPlanet class is the next step.

1.1.3.1 delete_items() function of spaceships is called to delete all the projectiles owned by each enemy spaceship.

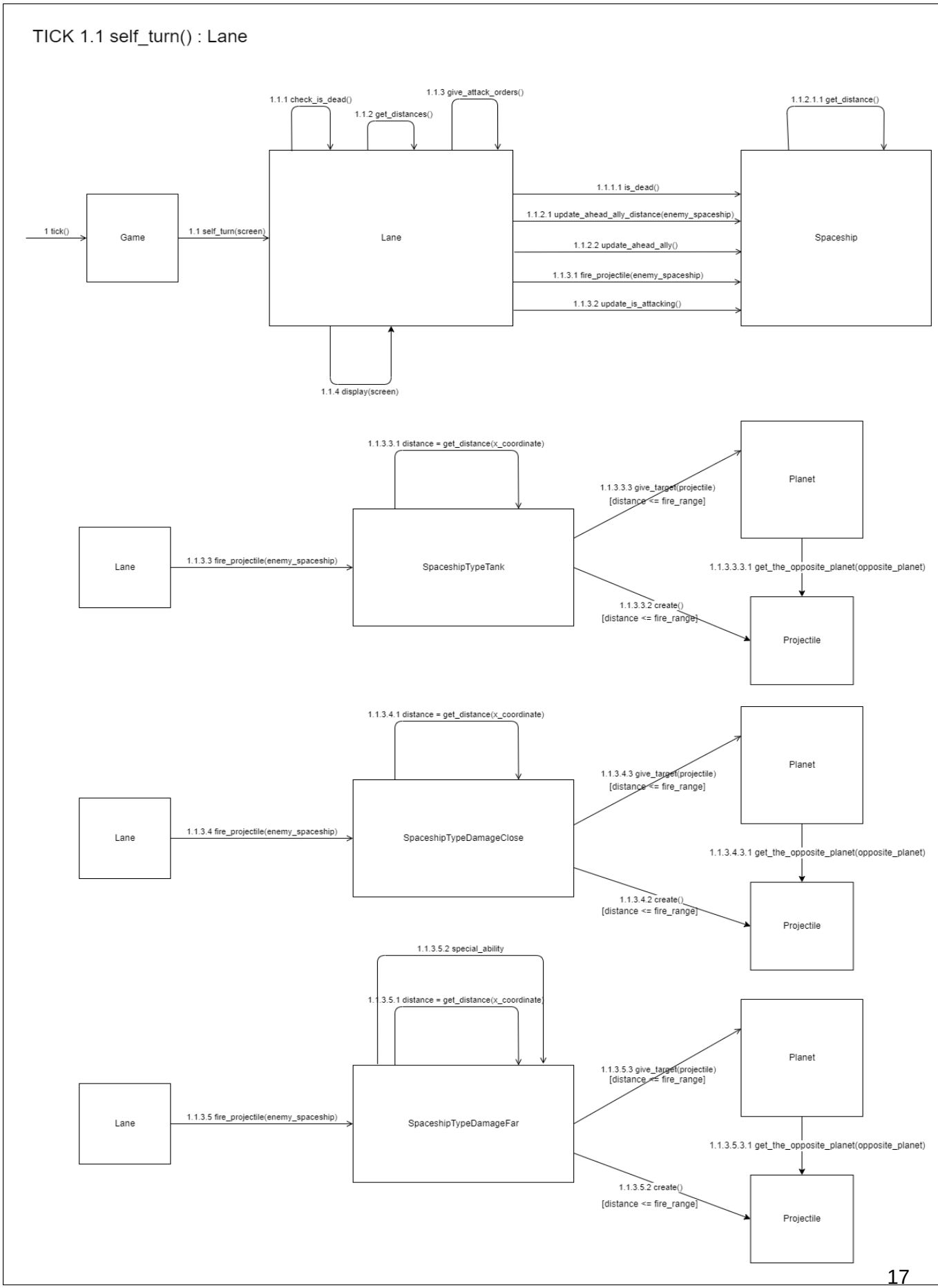
1.1.3.1.1 delete() function is called to delete the projectiles.

1.1.3.2 delete() function is called to delete each enemy spaceship after all of its projectiles are deleted.

1.1.4 After deleting all the spaceship and projectile class instances belonging to the enemy, the EnemyPlanet class itself is deleted with the delete() function.

1.1.5 delete() function is called to delete Lane instances.

TICK


TICK 1.1 self_turn() : Lane
1.1.1 check_is_dead()
1.1.2 get_distances()
1.1.3 give_attack_orders()
1.1.2.1.1 get_distance()
1 tick()
Game
1.1 self_turn(screen)
Lane
1.1.1.1 is_dead()
1.1.2.1 update_ahead_ally_distance(enemy_spaceship)
1.1.2.2 update_ahead_ally()
1.1.3.1 fire_projectile(enemy_spaceship)
1.1.3.2 update_is_attacking()
Spaceship
1.1.4 display(screen)
1.1.3.3.1 distance = get_distance(x_coordinate)
Lane
1.1.3.3 fire_projectile(enemy_spaceship)
SpaceshipTypeTank
1.1.3.3.3 give_target(projectile)
[distance <= fire_range]
Planet
1.1.3.3.3.1 get_the_opposite_planet(opposite_planet)
1.1.3.3.2 create()
[distance <= fire_range]
Projectile
1.1.3.4.1 distance = get_distance(x_coordinate)
Lane
1.1.3.4 fire_projectile(enemy_spaceship)
SpaceshipTypeDamageClose
1.1.3.4.3 give_target(projectile)
[distance <= fire_range]
Planet
1.1.3.4.3.1 get_the_opposite_planet(opposite_planet)
1.1.3.4.2 create()
[distance <= fire_range]
Projectile
1.1.3.5.2 special_ability
1.1.3.5.1 distance = get_distance(x_coordinate)
Lane
1.1.3.5 fire_projectile(enemy_spaceship)
SpaceshipTypeDamageFar
1.1.3.5.3 give_target(projectile)
[distance <= fire_range]
Planet
1.1.3.5.3.1 get_the_opposite_planet(opposite_planet)
1.1.3.5.2 create()
[distance <= fire_range]
Projectile

1 tick() function is called in order to process all the things done by the CPU.

1.1 The self_turn() function of the Lane is called. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.1.1 The check_is_dead() function is called. This function checks if any Spaceship instance is dead and if so it deletes that spaceship instance from the spaceship list it belongs to.

1.1.1.1 The is_dead() function of the Spaceship class is called for each leading spaceship to check if they died at that frame. If the spaceship is dead, this function returns True.

1.1.2 The get_distances() function of the Lane is called in order to give ally spaceships distance to the other ally spaceships.

1.1.2.1 The update_ahead_ally_distance(ally_spaceship) function is called to get the distance information of the spaceships.

1.1.2.1.1 The get_distance() function is called to get the updated distance of the spaceships.

1.1.3 give_attack_orders() function is called. The goal of this function is to make spaceships fire projectiles automatically if there is any enemy spaceship within its range.

1.1.3.1 The fire_projectile(enemy_spaceship) function is called to make spaceships create a new projectile.

1.1.3.2 The update_is_attacking() function is called to change the boolean is_attacking to False.

1.1.3.3 The fire_projectile(enemy_spaceship) function is called to make spaceships create a new projectile.

1.1.3.3.1 The get_distance function(x_coordiate) of the enemy spaceship is called.

1.1.3.3.2 The create() function creates the projectile.

1.1.3.3.3 The give_target(projectile) function is called to make Planet give the target coordinates to the projectile.

1.1.3.3.3.1 The get_the_opposite_planet(opposite_planet) function is called to give the projectile the coordinates of the closest enemy.

1.1.3.4 The fire_projectile(enemy_spaceship) function is called to make spaceships create a new projectile.

1.1.3.4.1 The get_distance function(x_coordiate) of the enemy spaceship is called.

1.1.3.4.2 The create() function creates the projectile.

1.1.3.4.3 The give_target(projectile) function is called to make Planet give the target coordinates to the projectile.

1.1.3.4.3.1 The get_the_opposite_planet(opposite_planet) function is called to give the projectile the coordinates of the closest enemy.

1.1.3.5 The fire_projectile(enemy_spaceship) function is called to make spaceships create a new projectile.

1.1.3.5.1 The get_distance function(x_coordiate) of the enemy spaceship is called.

1.1.3.5.2 The special_ability() function is called.

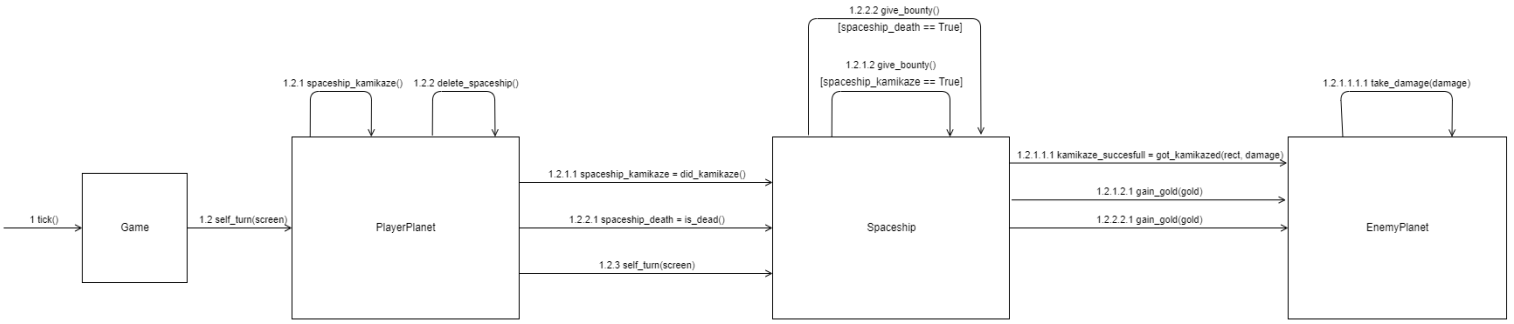
1.1.3.5.3 The create() function creates the projectile.

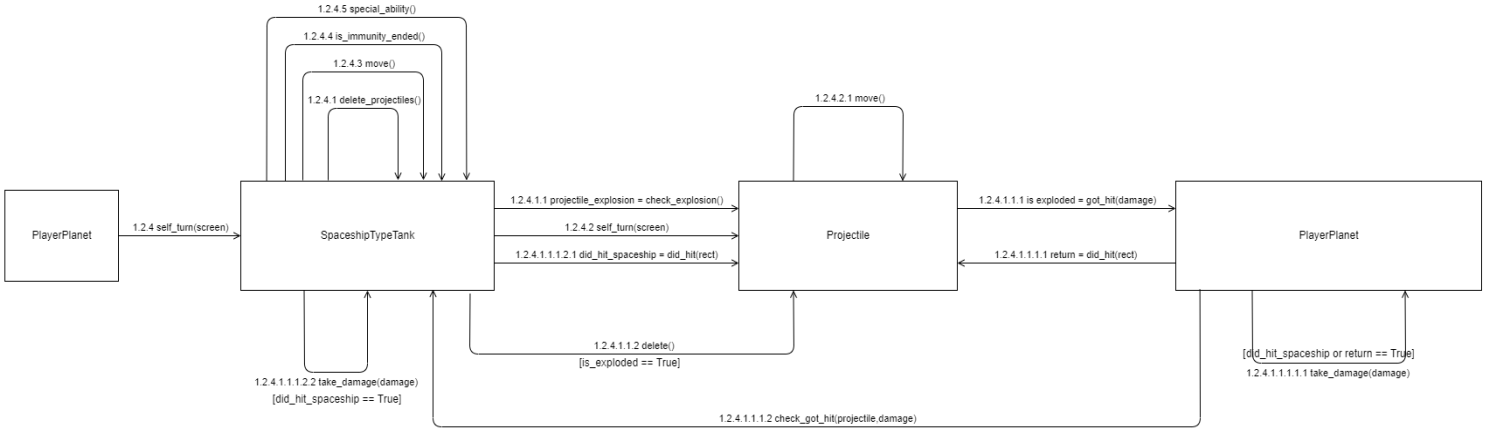
1.1.3.5.4 The give_target(projectile) function is called to make Planet give the target coordinates to the projectile.

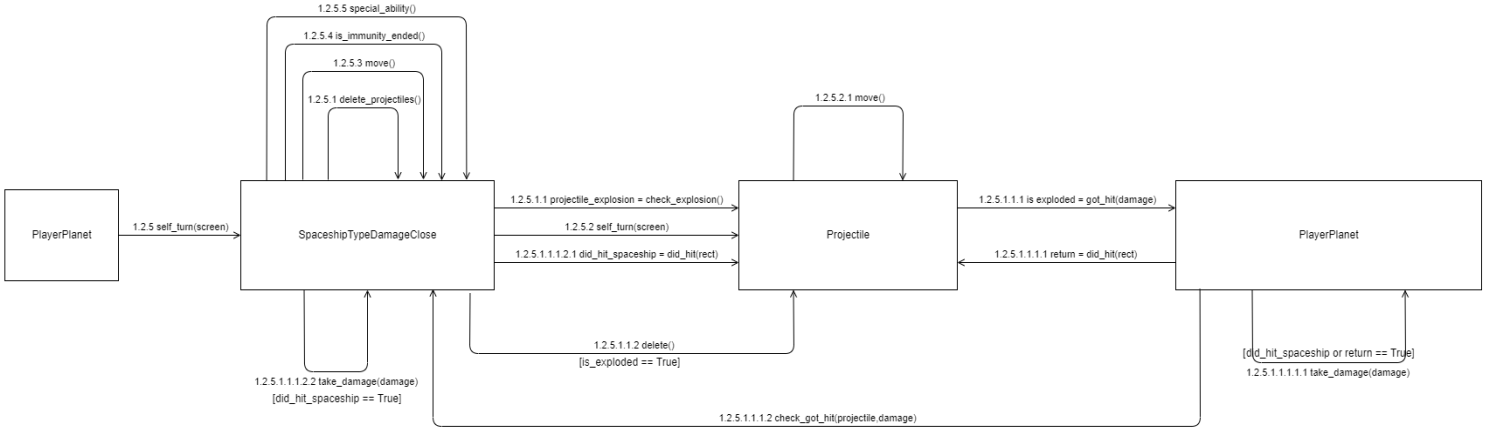
1.1.3.5.4.1 The get_the_opposite_planet(opposite_planet) function is called to give the projectile the coordinates of the closest enemy.

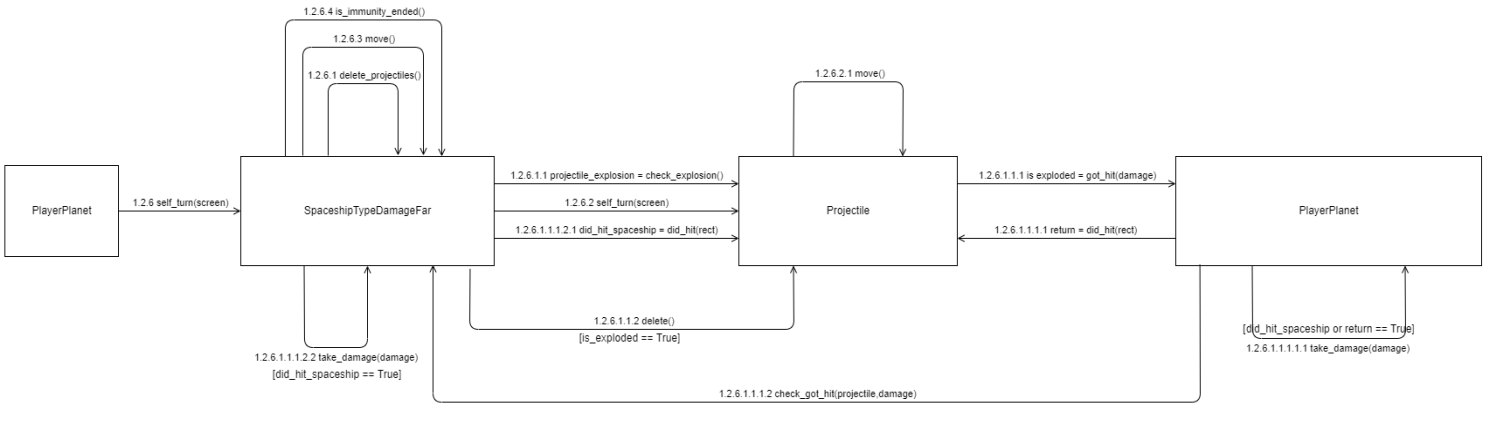
1.1.4 The display() function of the Lane class is called in order to display the lane on the screen.

TICK 1.2 self_turn() : PlayerPlanet









1.2 The self_turn() function of the PlayerPlanet class is called.

1.2.1 The spaceship_kamikaze() function is called to check whether a player spaceship is kamikazed the enemy planet, and did any damage to it.

1.2.1.1 The did_kamikaze() function of the spaceship is called and it returns the boolean spaceship_kamikaze variable.

1.2.1.1.1 Inside the did_kamikaze() function, got_kamikazed(rect, damage) function of the EnemyPlanet class is called, and it returns the boolean kamikaze_succesfull variable. If this variable equals True, health of the spaceship is set to zero. The got_kamikazed(rect, damage) function checks for rectangle collision and calls the take_damage(damage) function of player planet if a collision occurs.

1.2.1.1.1.1 The take_damage(damage) function decreases the player planets health value according to the damage done from the kamikaze attack.

1.2.1.2 If the returned value of spaceship_kamikaze equals True, give_bounty() function of the spaceship is called. This function gives the bounty value of the spaceship to the opponent planet. Inside the give_bounty() function, gain_gold(gold) function of the PlayerPlanet is called.

1.2.1.2.1 The give_gold(gold) function increases the gold amount of player planet by the bounty value of spaceship.

1.2.2 The delete_spaceship() function of the PlayerPlanet is called to delete any spaceship that have zero health.

1.2.2.1 The is_dead() function of the spaceships of enemy planet is called, and it returns a boolean spaceship_death variable. This function checks whether the spaceship is dead or not.

1.2.2.2 If the spaceship_death variable equals True, give_bounty() function of the spaceship is called. This function gives the bounty value of the spaceship to the opponent planet. Inside the give_bounty() function, gain_gold(gold) function of the PlayerPlanet is called.

1.2.2.2.1 The give_gold(gold) function increases the gold amount of player planet by the bounty value of spaceship.

1.2.3 The self_turn() of the spaceship class is called in order for spaceships to do their functions in the frame.

1.2.4 The self_turn() function of the SpaceshipTypeTank class is called for all the spaceships that EnemyPlanet has. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.2.4.1 The delete_projectiles() function of the SpaceshipTypeTank class loops through all the projectiles it has and deletes any exploded projectiles. In order to do that, the function calls check_explosion() function of the Projectile class.

1.2.4.1.1 The check_explosion() function of the Projectile class calls the got_hit() function of the PlayerPlanet class with the projectile's damage attribute as the input parameter.

1.2.4.1.1.1 Inside the got_hit(enemy_projectile, damage) function, the did_hit() function of the Projectile class is called.

1.2.4.1.1.1.1 In did_hit() function, pygame collision check is made.

1.2.4.1.1.1.1.1 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the PlayerPlanet class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the PlayerPlanet's health by the amount of projectile's damage it gets hit by.

1.2.4.1.1.1.2 If the did_hit() function returns False, for every spaceship that PlayerPlanet has, the check_got_hit() function of the Spaceship class is called.

1.2.4.1.1.1.2.1 In did_hit() function, pygame collision check is made.

1.2.4.1.1.1.2.2 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the SpaceshipTypeTank class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the spaceship's health by the amount of projectile’s damage it gets hit by.

1.2.4.1.2 The delete() function of the Projectile class is called to delete the projectile which has already exploded and inflicted damage to other objects and the projectile is deleted.

1.2.4.2 The self_turn() function of the Projectile class is called for every instance of the class. This function calls the move() and super().display() functions of the projectiles.

1.2.4.2.1 The move() function of the Projectile class is called to make projectiles move.

1.2.4.2.2 The display() function of the parent class is called and the projectile gets displayed on the screen.

1.2.4.3 The move() function of the SpaceshipTypeTank class is called to make projectiles move.

1.2.4.4 The is_immunity_ended() function of the Spaceship class is called. This function checks if immunity time has ended and if immunity time has ended, it changes is_immune attribute to False which makes spaceships vulnerable to damage. It also records the immunity_end_time. This time is used to calculate when skill will be able to be used again.

1.2.4.5 The special_ability() function of the SpaceshipTypeTank is called.

1.2.4.6 The display() function of the parent class is called and the spaceships get displayed on the screen.

1.2.5 The self_turn() function of the SpaceshipTypeDamageClose class is called for all the spaceships that EnemyPlanet has. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.2.5.1 The delete_projectiles() function of the SpaceshipTypeDamageClose class loops through all the projectiles it has and deletes any exploded projectiles. In order to do that, the function calls check_explosion() function of the Projectile class.

1.2.5.1.1 The check_explosion() function of the Projectile class calls the got_hit() function of the PlayerPlanet class with the projectile's damage attribute as the input parameter.

1.2.5.1.1.1 Inside the got_hit(enemy_projectile, damage) function, the did_hit() function of the Projectile class is called.

1.2.5.1.1.1.1 In did_hit() function, pygame collision check is made.

1.2.5.1.1.1.1.1 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the PlayerPlanet class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the PlayerPlanet's health by the amount of projectile's damage it gets hit by.

1.2.5.1.1.1.2 If the did_hit() function returns False, for every spaceship that PlayerPlanet has, the check_got_hit() function of the Spaceship class is called.

1.2.5.1.1.1.2.1 In did_hit() function, pygame collision check is made.

1.2.5.1.1.1.2.2 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the SpaceshipTypeDamageClose class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the spaceship's health by the amount of projectile's damage it gets hit by.

1.2.5.1.2 The delete() function of the Projectile class is called to delete the projectile which has already exploded and inflicted damage to other objects and the projectile is deleted.

1.2.5.2 The self_turn() function of the Projectile class is called for every instance of the class. This function calls the move() and super().display() functions of the projectiles.

1.2.5.2.1 The move() function of the Projectile class is called to make projectiles move.

1.2.5.2.2 The display() function of the parent class is called and the projectile gets displayed on the screen.

1.2.5.3 The move() function of the SpaceshipTypeDamageClose class is called to make projectiles move.

1.2.5.4 The is_immunity_ended() function of the Spaceship class is called. This function checks if immunity time has ended and if immunity time has ended, it changes is_immune attribute to False which makes spaceships vulnerable to damage. It also records the immunity_end_time. This time is used to calculate when skill will be able to be used again.

1.2.5.5 The special_ability() function of the SpaceshipTypeDamageClose is called.

1.2.5.6 The display() function of the parent class is called and the spaceships get displayed on the screen.

1.2.6 The self_turn() function of the SpaceshipTypeDamageFar class is called for all the spaceships that EnemyPlanet has. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.2.6.1 The delete_projectiles() function of the SpaceshipTypeDamageFar class loops through all the projectiles it has and deletes any exploded projectiles. In order to do that, the function calls check_explosion() function of the Projectile class.

1.2.6.1.1 The check_explosion() function of the Projectile class calls the got_hit() function of the PlayerPlanet class with the projectile's damage attribute as the input parameter.

1.2.6.1.1.1 Inside the got_hit(enemy_projectile, damage) function, the did_hit() function of the Projectile class is called.

1.2.6.1.1.1.1 In did_hit() function, pygame collision check is made.

1.2.6.1.1.1.1.1 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the PlayerPlanet class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage)

function decreases the PlayerPlanet’s health by the amount of projectile’s damage it gets hit by.

1.2.6.1.1.1.2 If the did_hit() function returns False, for every spaceship that PlayerPlanet has, the check_got_hit() function of the Spaceship class is called.

1.2.6.1.1.1.2.1 In did_hit() function, pygame collision check is made.

1.2.6.1.1.1.2.2 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the SpaceshipTypeDamageFar class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the spaceship's health by the amount of projectile’s damage it gets hit by.

1.2.6.1.2 The delete() function of the Projectile class is called to delete the projectile which has already exploded and inflicted damage to other objects and the projectile is deleted.

1.2.6.2 The self_turn() function of the Projectile class is called for every instance of the class. This function calls the move() and super().display() functions of the projectiles.

1.2.6.2.1 The move() function of the Projectile class is called to make projectiles move.

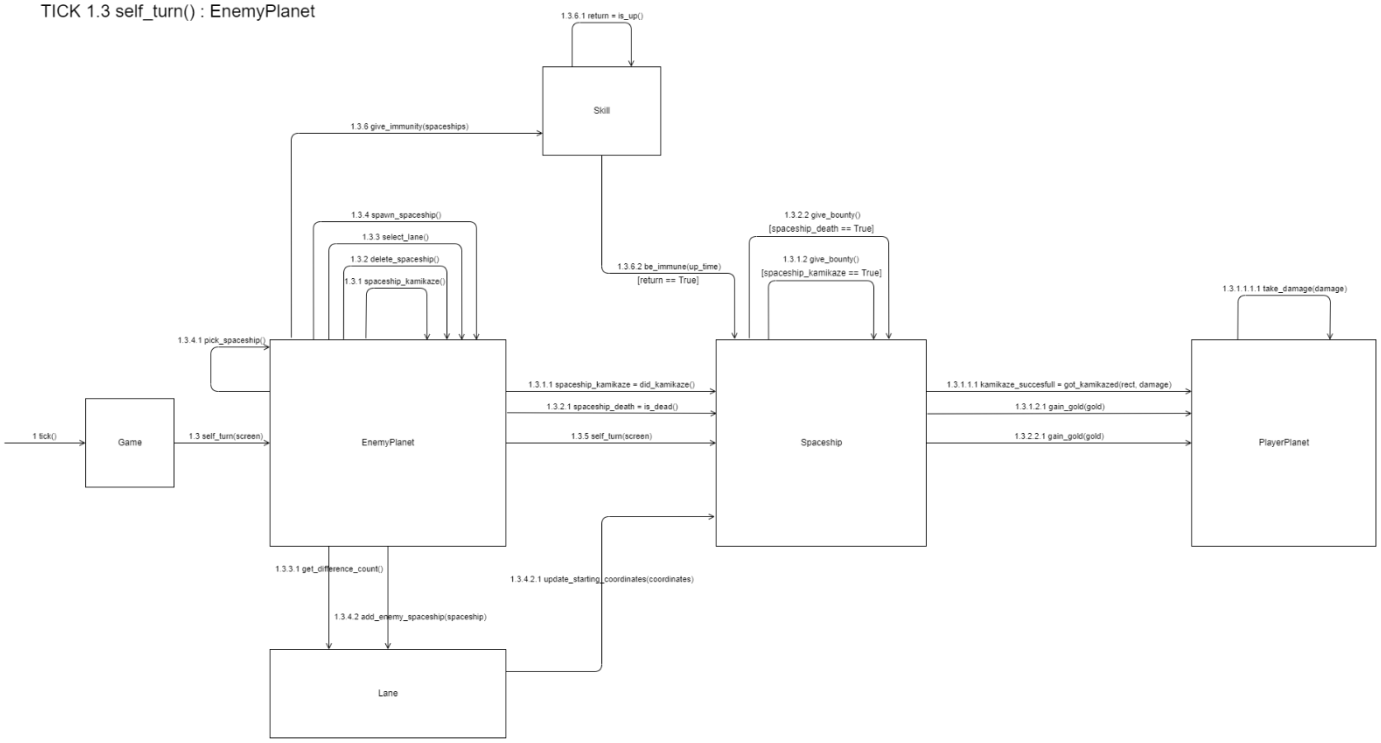
1.2.6.2.2 The display() function of the parent class is called and the projectile gets displayed on the screen.

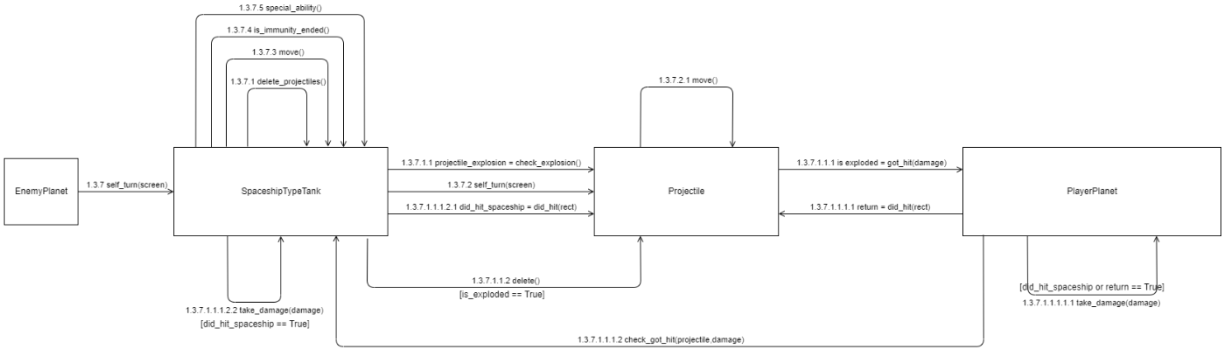
1.2.6.3 The move() function of the SpaceshipTypeDamageFar class is called to make projectiles move.

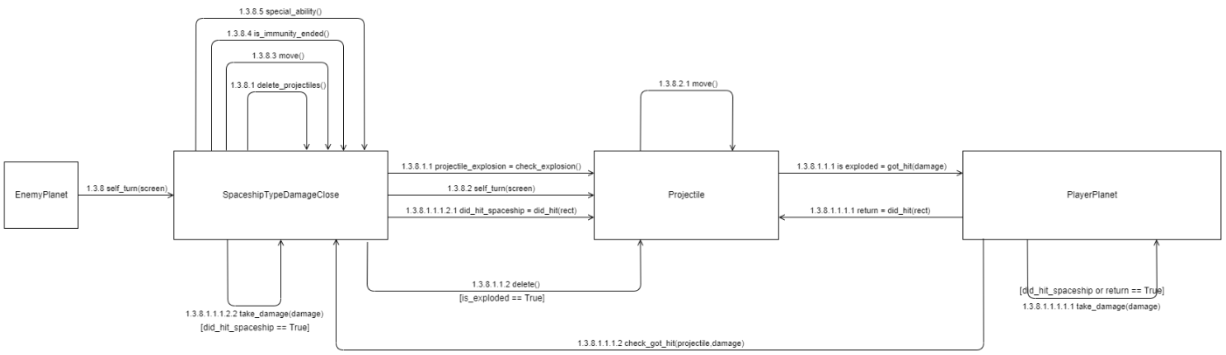
1.2.6.4 The is_immunity_ended() function of the Spaceship class is called. This function checks if immunity time has ended and if immunity time has ended, it changes is_immune attribute to False which makes spaceships vulnerable to damage. It also records the immunity_end_time. This time is used to calculate when skill will be able to be used again.

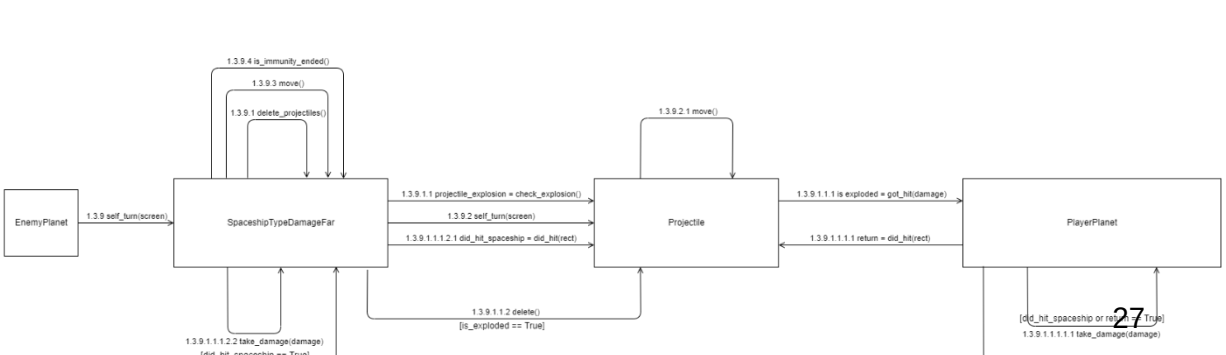
1.2.6.5 The display() function of the parent class is called and the spaceships get displayed on the screen.

TICK 1.3 self_turn() : EnemyPlanet









1.3 The self_turn() function of the EnemyPlanet is called. This function operates enemy players functionality and it checks and updates the attributes of the entity at each frame.

1.3.1 The spaceship_kamikaze() function is called to check whether a player spaceship is kamikazed the enemy planet, and did any damage to it.

1.3.1.1 The did_kamikaze() function of the spaceship is called and it returns the boolean spaceship_kamikaze variable.

1.3.1.1.1 Inside the did_kamikaze() function, got_kamikazed() function of the PlayerPlanet class is called, and it returns the boolean kamikaze_succesfull variable. If this variable equals True, health of the spaceship is set to zero. The got_kamikazed() function checks for rectangle collision and calls the take_damage(damage) function of player planet if a collision occurs.

1.3.1.1.1.1 The take_damage(damage) function decreases the player planets health value according to the damage done from the kamikaze attack.

1.3.1.2 If the returned value of spaceship_kamikaze equals True, give_bounty() function of the spaceship is called. This function gives the bounty value of the spaceship to the opponent planet. Inside the give_bounty() function, gain_gold(gold) function of the PlayerPlanet is called.

1.3.1.2.1 The give_gold(gold) function increases the gold amount of player planet by the bounty value of spaceship.

1.3.2 The delete_spaceship() function of the EnemyPlanet is called to delete any spaceship that have zero health.

1.3.2.1 The is_dead() function of the spaceships of enemy planet is called, and it returns a boolean spaceship_death variable. This function checks whether the spaceship is dead or not.

1.3.2.2 If the spaceship_death variable equals True, give_bounty() function of the spaceship is called. This function gives the bounty value of the spaceship to the opponent planet. Inside the give_bounty() function, gain_gold(gold) function of the PlayerPlanet is called.

1.3.2.2.1 The give_gold(gold) function increases the gold amount of player planet by the bounty value of spaceship.

1.3.3 The select_lane() function of the EnemyPlanet is called to select the lane which enemy player will spawn it's spaceships.

1.3.3.1 The get_difference_count() function of the Lane class is called. This function checks for the difference in numbers of enemy and player spaceships within lanes. The lane with the maximum difference in numbers is selected to spawn enemy spaceships.

1.3.4 The spawn_spaceship() function of the EnemyPlanet class is called, in order to make enemy planet spawn spaceship and attack it's opponent.

1.3.4.1 The pick_spaceship() function is called to randomly chose a spaceship to spawn.

1.3.4.2 The add_enemy_spaceship(spaceship) function of lane is called to add the newly spawned spaceship to the lane class's list storing enemy spaceships.

1.3.4.2.1 The update_starting_coordinates(coordinates) function of spaceship is called to give spaceship its starting coordinates.

1.3.5 The self_turn() function of the spaceship is called to execute the functionality of the enemy planets' spaceships in each frame. All the functions up until 1.3.5.4 are same for all 3 types of spaceships, but 1.3.5.5 is only for SpaceshipTypeDamageClose and SpaceshipTypeTank.

1.3.6 The give_immunity() function of the skill is called with spaceships as parameters to make enemy planet use its skill and give immunity to its spaceships.

1.3.6.1 The is_up() function of the skill is called within the give_immunity(), to check whether skill is ready to use, and it returns a boolean value

1.3.6.2 If the returned value is True, be_immune(up_time) function of the Spaceship is called to grant immunity to all enemy spaceships and update the immunity ending time.

1.3.7 The self_turn() function of the SpaceshipTypeTank class is called for all the spaceships that EnemyPlanet has. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.3.7.1 The delete_projectiles() function of the SpaceshipTypeTank class loops through all the projectiles it has and deletes any exploded projectiles. In order to do that, the function calls check_explosion() function of the Projectile class.

1.3.7.1.1 The check_explosion() function of the Projectile class calls the got_hit() function of the PlayerPlanet class with the projectile's damage attribute as the input parameter.

1.3.7.1.1.1 Inside the got_hit(enemy_projectile, damage) function, the did_hit() function of the Projectile class is called.

1.3.7.1.1.1.1 In did_hit() function, pygame collision check is made.

1.3.7.1.1.1.1.1 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the PlayerPlanet class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the PlayerPlanet's health by the amount of projectile's damage it gets hit by.

1.3.7.1.1.1.2 If the did_hit() function returns False, for every spaceship that PlayerPlanet has, the check_got_hit() function of the Spaceship class is called.

1.3.7.1.1.1.2.1 In did_hit() function, pygame collision check is made.

1.3.7.1.1.1.2.2 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the SpaceshipTypeTank class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the spaceship's health by the amount of projectile's damage it gets hit by.

1.3.7.1.2 The delete() function of the Projectile class is called to delete the projectile which has already exploded and inflicted damage to other objects and the projectile is deleted.

1.3.7.2 The self_turn() function of the Projectile class is called for every instance of the class. This function calls the move() and super().display() functions of the projectiles.

1.3.7.2.1 The move() function of the Projectile class is called to make projectiles move.

1.3.7.2.2 The display() function of the parent class is called and the projectile gets displayed on the screen.

1.3.7.3 The move() function of the SpaceshipTypeTank class is called to make projectiles move.

1.3.7.4 The is_immunity_ended() function of the Spaceship class is called. This function checks if immunity time has ended and if immunity time has ended, it changes is_immune attribute to False which makes spaceships vulnerable to damage. It also records the immunity_end_time. This time is used to calculate when skill will be able to be used again.

1.3.7.5 The special_ability() function of the SpaceshipTypeTank is called.

1.3.7.6 The display() function of the parent class is called and the spaceships get displayed on the screen.

1.3.8 The self_turn() function of the SpaceshipTypeDamageClose class is called for all the spaceships that EnemyPlanet has. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.3.8.1 The delete_projectiles() function of the SpaceshipTypeDamageClose class loops through all the projectiles it has and deletes any exploded projectiles. In order to do that, the function calls check_explosion() function of the Projectile class.

1.3.8.1.1 The check_explosion() function of the Projectile class calls the got_hit() function of the PlayerPlanet class with the projectile's damage attribute as the input parameter.

1.3.8.1.1.1 Inside the got_hit(enemy_projectile, damage) function, the did_hit() function of the Projectile class is called.

1.3.8.1.1.1.1 In did_hit() function, pygame collision check is made.

1.3.8.1.1.1.1.1 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the PlayerPlanet class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the PlayerPlanet's health by the amount of projectile's damage it gets hit by.

1.3.8.1.1.1.2 If the did_hit() function returns False, for every spaceship that PlayerPlanet has, the check_got_hit() function of the Spaceship class is called.

1.3.8.1.1.1.2.1 In did_hit() function, pygame collision check is made.

1.3.8.1.1.1.2.2 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the SpaceshipTypeDamageClose class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the spaceship's health by the amount of projectile's damage it gets hit by.

1.3.8.1.2 The delete() function of the Projectile class is called to delete the projectile which has already exploded and inflicted damage to other objects and the projectile is deleted.

1.3.8.2 The self_turn() function of the Projectile class is called for every instance of the class. This function calls the move() and super().display() functions of the projectiles.

1.3.8.2.1 The move() function of the Projectile class is called to make projectiles move.

1.3.8.2.2 The display() function of the parent class is called and the projectile gets displayed on the screen.

1.3.8.3 The move() function of the SpaceshipTypeDamageClose class is called to make projectiles move.

1.3.8.4 The is_immunity_ended() function of the Spaceship class is called. This function checks if immunity time has ended and if immunity time has ended, it changes is_immune attribute to False which makes spaceships vulnerable to damage. It also records the immunity_end_time. This time is used to calculate when skill will be able to be used again.

1.3.8.5 The special_ability() function of the SpaceshipTypeDamageClose is called.

1.3.8.6 The display() function of the parent class is called and the spaceships get displayed on the screen.

1.3.9 The self_turn() function of the SpaceshipTypeDamageFar class is called for all the spaceships that EnemyPlanet has. This function does more than just displaying the entity, it basically checks and updates the attributes of the entity at each instance.

1.3.9.1 The delete_projectiles() function of the SpaceshipTypeDamageFar class loops through all the projectiles it has and deletes any exploded projectiles. In order to do that, the function calls check_explosion() function of the Projectile class.

1.3.9.1.1 The check_explosion() function of the Projectile class calls the got_hit() function of the PlayerPlanet class with the projectile's damage attribute as the input parameter.

1.3.9.1.1.1 Inside the got_hit(enemy_projectile, damage) function, the did_hit() function of the Projectile class is called.

1.3.9.1.1.1.1 In did_hit() function, pygame collision check is made.

1.3.9.1.1.1.1.1 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the PlayerPlanet class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the PlayerPlanet's health by the amount of projectile's damage it gets hit by.

1.3.9.1.1.1.2 If the did_hit() function returns False, for every spaceship that PlayerPlanet has, the check_got_hit() function of the Spaceship class is called.

1.3.9.1.1.1.2.1 In did_hit() function, pygame collision check is made.

1.3.9.1.1.1.2.2 If the did_hit() function returns True, the condition for taking damage is met and the take_damage(damage) function of the SpaceshipTypeDamageFar class is called with the input of the damage attribute of the projectile that exploded. The take_damage(damage) function decreases the spaceship's health by the amount of projectile’s damage it gets hit by.

1.3.9.1.2 The delete() function of the Projectile class is called to delete the projectile which has already exploded and inflicted damage to other objects and the projectile is deleted.

1.3.9.2 The self_turn() function of the Projectile class is called for every instance of the class. This function calls the move() and super().display() functions of the projectiles.

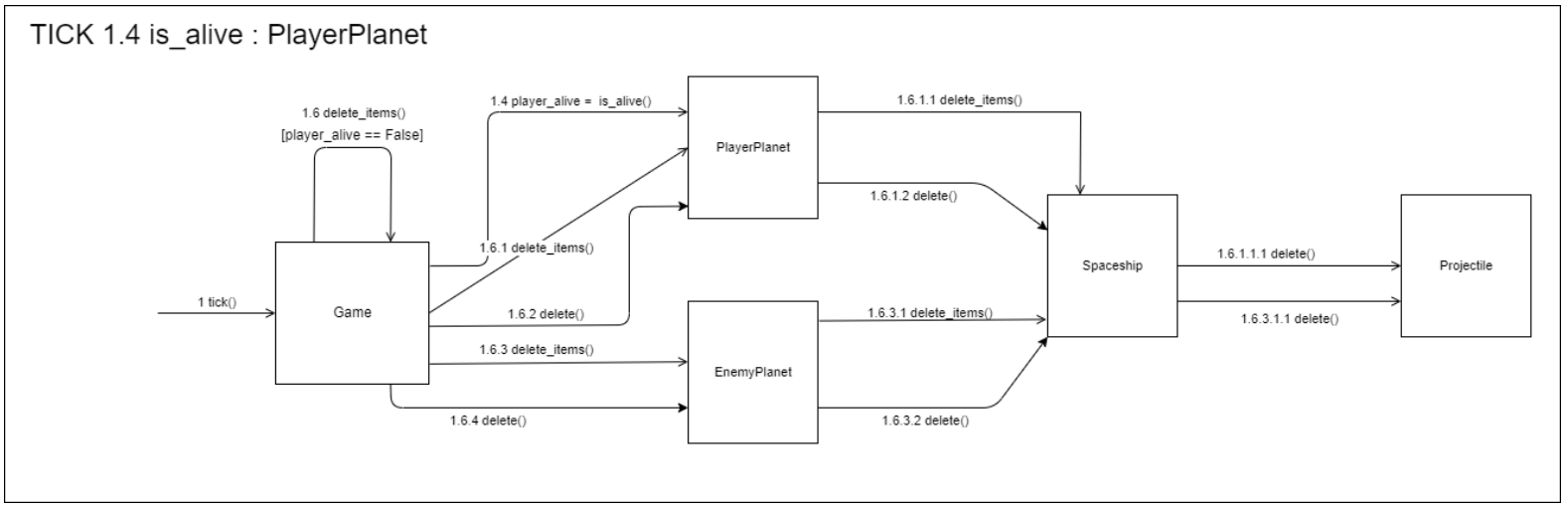
1.3.9.2.1 The move() function of the Projectile class is called to make projectiles move.

1.3.9.2.2 The display() function of the parent class is called and the projectile gets displayed on the screen.

1.3.9.3 The move() function of the SpaceshipTypeDamageFar class is called to make projectiles move.

1.3.9.4 The is_immunity_ended() function of the Spaceship class is called. This function checks if immunity time has ended and if immunity time has ended, it changes is_immune attribute to False which makes spaceships vulnerable to damage. It also records the immunity_end_time. This time is used to calculate when skill will be able to be used again.

1.3.9.5 The display() function of the parent class is called and the spaceships get displayed on the screen.



1.4 The is_alive() function is called at each instance to check if the amount of health PlayerPlanet has is more than zero. When the PlayerPlanet is dead, player_alive boolean becomes False.

1.6 If the player_alive Boolean is False, that means the game is over and all the class instances must be deleted. Thus, delete_items() function is called inside the Game object.

1.6.1 First, the delete_items() function is called in order to start deleting the PlayerPlanet class and all the spaceship class instances owned by the player. Since we cannot delete the rest of the classes if the PlayerPlanet class is deleted first, it is deleted later.

1.6.1.1 delete_items() function is called to delete all the projectiles owned by each player spaceship. Same reasoning with why we firstly use delete_items for deleting Planets.

1.6.1.1.1 delete() function is called to delete the projectiles.

1.6.1.2 delete() function is called to delete each player spaceship after all the projectiles of player spaceships are deleted.

1.6.2 After deleting all the spaceship and projectile class instances belonging to the player, the PlayerPlanet class itself is deleted with the delete() function.

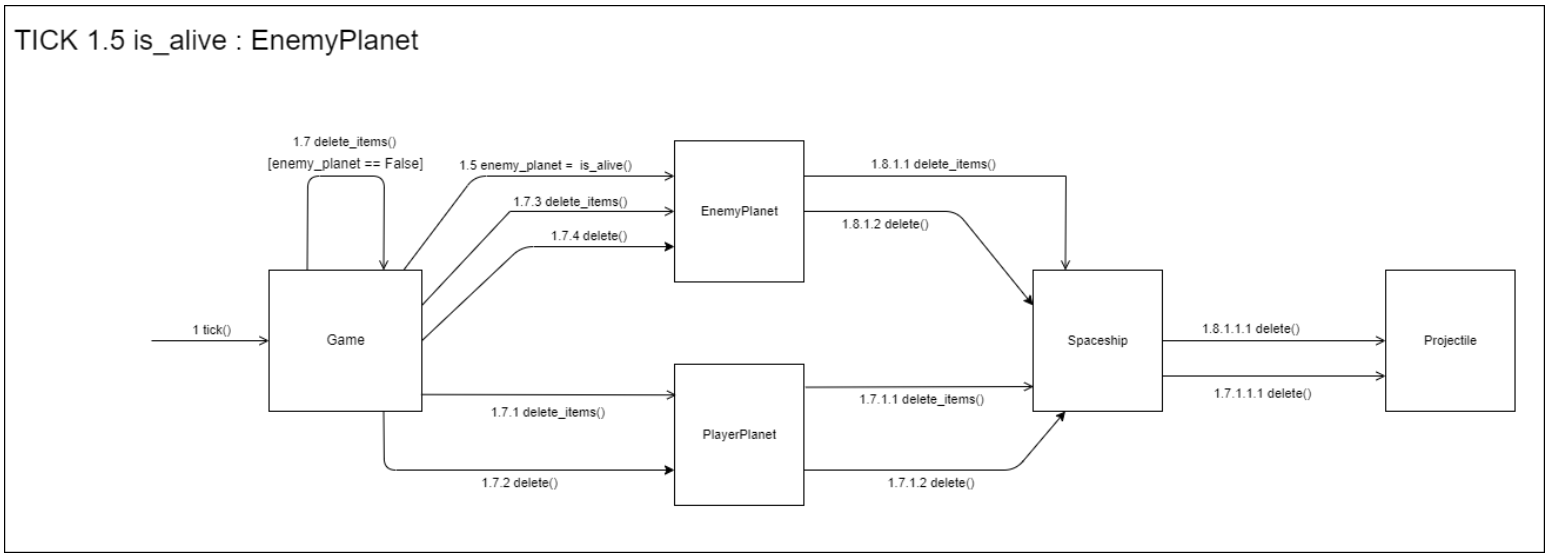
1.6.3 First, the delete_items() function is called in order to start deleting the EnemyPlanet class and all the spaceship class instances owned by the enemy. Since we cannot delete the rest of the instances if the EnemyPlanet class is deleted first, it is deleted at the later steps.

1.6.3.1 The delete_items() function is called to delete all the projectiles owned by each enemy spaceship.

1.6.3.1.1 The delete() function is called to delete the projectiles.

1.6.3.2 The delete() function is called to delete each enemy spaceship after all theprojectiles of enemy spaceships are deleted.

1.6.4 After deleting all the spaceship and projectile class instances belonging to the enemy, the EnemyPlanet class itself is deleted with the delete() function



1.5 The is_alive() function is called at each instance to check if the amount of health EnemyPlanet has is more than zero. When the EnemyPlanet is dead, enemy_alive boolean becomes False.

1.7 If the enemy_alive Boolean is False, that means the game is over and all the class instances must be deleted. Thus, delete_items() function is called inside the Game object.

1.7.1 First, the delete_items() function is called in order to start deleting the PlayerPlanet class and all the spaceship class instances owned by the player. Since we cannot delete the rest of the classes if the PlayerPlanet class is deleted first, it is deleted later.

1.7.1.1 delete_items() function is called to delete all the projectiles owned by each

player spaceship. Same reasoning with why we firstly use delete_items for deleting Planets.

1.7.1.1.1 delete() function is called to delete the projectiles.

1.7.1.2 delete() function is called to delete each player spaceship after all the projectiles player spaceships are deleted.

1.7.2 After deleting all the spaceship and projectile class instances belonging to the player, the PlayerPlanet class itself is deleted with the delete() function.

1.7.3 First, the delete_items() function is called in order to start deleting the EnemyPlanet class and all the spaceship class instances owned by the enemy. Since we cannot delete the rest of the instances if the EnemyPlanet class is deleted first, it is deleted at the later steps.

1.7.3.1 The delete_items() function is called to delete all the projectiles owned by each

enemy spaceship.

1.7.3.1.1 The delete() function is called to delete the projectiles.

1.7.3.2 The delete() function is called to delete each enemy spaceship after all theprojectiles of enemy spaceships are deleted.

1.7.4 After deleting all the spaceship and projectile class instances belonging to the enemy, the EnemyPlanet class itself is deleted with the delete() function.

User Documentation

Player selects the lane that he/she wants to spawn a spaceship. The selection is done by **arrow up** or **arrow down** buttons. The selected lane will be reflected on the screen.

After there is a selected lane, players can choose the type of spaceship that they want to summon. Button "**a**" spawns Tank Spaceships, button "**s**" spawns Damage Close Spaceships and button "**d**" spawns Damage Far Spaceships. At the instance of pushing the button, if prerequisites are satisfied -such as enough gold-, selected type of spaceship appears on the selected lane. After players spawn a spaceship, in order to spawn a new spaceship, they must wait at least 2 seconds.

To use the skill, players push the **space** button. If skill is available, all the spaceships that players currently have become immune to damage for 7.5 seconds. This immunity is visually reflected as a barrier around the spaceships. To use the skill again, players must wait 30 seconds.

To quit the game before game ends, players can press the button "**f**" or can click the close button on the edge of the window.

If game ends normally (i.e., players win or lose), end game screen becomes visible for 3 seconds displaying the result of the game. After that, game quits automatically.