SPRINT 1 – Team No. 1

- Install unity on all computers
  - ⇨ We installed unity version 2021.3.11f1
- Get familiar with unity interface
- Attend the weekly meeting
  - ⇨ Monday at 8:30 PM
- Explain your status on the task
  - ⇨ Each task was assigned to one teammate
- Set up our GitHub accounts ⇨ Assigned to Denisa
  - Create repository
  - Give access to all teammates
    - ⇨ Link to GitHub: https://github.com/denisapredescu/Mad-Drake
- Set up our Trello board ⇨ Assigned to Denisa
  - ⇨ Link to Trello workspace: https://trello.com/w/workspace31329919
- Create project ⇨ Assigned to Patricia
  - Project name: Mad Drake
  - Create the necessary folders
- Create camera and room ⇨ Assigned to Patricia
  - Camera to be "responsive" on different aspect ratios.
    - ⇨ **WORK IN PROGRESS**
    - ⇨ Camera is fixed, because the responsive camera is not a priority for the prototype and testing.
- Create HUD ⇨ Assigned to Amer
  - Display information about the player (health status, collected gold)

⇨ **DONE**

⇨ Added Methods:

- SetHealth(int health)
- SetMaxHealth(int health)
- TakeDamage(int damage)
- AddGold()

⇨ SetHealth(int health) modifies the health slidebar to represent the amount and color corresponding to the "health" parameter

⇨ SetMaxHealth(int health) initializes the health bar to its maximum value, which will be equal to the "health" parameter.

⇨ TakeDamage(int damage) decreases the current health of the player by the specified damage parameter

⇨ AddGold() increases the current gold amount collected by the player by 1

→ The new script that contains the SetHealth() and SetMaxHealth() methods is called "HealthBar.cs" and is located in the Assets > Scripts folder.

→ The TakeDamage() and AddGold() methods have been added to the existing PlayerController script.

→ The overall change is that now the player has a HUD which displays the health bar and collected gold in the upper left side of the display.

- Create character object ⇨ Assigned to Marian
  - Create and initialize the player status variables such as health, gold, speed
  - Connect them with the HUD

    ⇨ **DONE**

⇨ Added a new script called "PlayerHealthController" in Assets > Scripts that deals with the player health logic and gold status

→ referenced the health bar and gold amount from HUD and connected them with the logic from the new script

→ added currentHealth, maxHealth, goldScore, playerIsDead variables

→ initialized maximum health and gold

→ moved TakeDamge and AddGold methods from PlayerController in the new script

→ added the GetPlayerDeadStatus method which with you can check if the player is currently dead from other scripts

→ in TakeDamage method added the dead status checking

- Create character movement ⇨ Assigned to Rareș
  - Create a movement in a 2D space not affected by gravity
  - Create a dash with a trail effect
  - Create the rotation of the gun to target the mouse position along with a gun template that has 2 gizmos, one as a pivot for rotation, the other one for firing point
  - Create a prefab to show how the character should be built in relation to the scripts

    ⇨ **DONE**

    ⇨ PlayerController handles the movement of the character, it is attached to the parent object of the character.

    → in update it moves the character up, down, left and right by reading the input, also uses the method RotatePlayer() to check if the character needs to change it's orientation. Furthermore, checks if space was pressed to start dashing

    → in fixed update it applies forces to create the dash effect

⇨ GunMotion handles the rotation of the gun used by the character, it is attached to the gun object (that is attached to a pivot object who is also attached to the character object), and uses the pivot and another object that has the coordinates of the firing point to calculate the rotation
→ in update are use two methods, CalculateDifferenceVector() calculates a vector used for rotation, RotateGun() uses that vector and applies the rotation

- Create rooms
  - Make rooms with all possible entries
  - Make different room centers

    ⇨ **NOT STARTED YET**

    ⇨ It is not a priority for the first prototype (it's a boring and repetitive task). We will leave this for the next sprint.
- Research how to do the procedural generation for the whole level using all types of rooms

    ⇨ **NOT STARTED YET**

    ⇨ This task is linked to the task above. It will be done in the next sprint.

"⇨" means DONE

"⇨" means WORK IN PROGRESS

"⇨" means NOT STARTED YET