•Topic of the project: the project is a combination two games – doodle jump and catch game.

•Program name: HSEss (HSE Student Simulator)

•Annotation:

The idea of this project is to combine two well-known and loved games in a pixel aesthetic and with our university life in mind. The functionality consists of the choice of a character (each with their own abilities that represent the members of the team) and subsequent choice of the game mode.

The main point of the catch game is to get as much “CODE: 0” (it refers to the “program exited with code 0” statement which shows that the program compiled successfully) as possible, while avoiding the “COMPILE ERROR” ones. The object is caught at the collision with the character. Good and bad objects are chosen with our programming experience in mind: errors decrease the score while “CODE 0”s increase it. The game time is limited; however, the limit is easily changeable through the Unity UI.

Every character in this game has special abilities. Vanya is faster than the others, Darya can skip one bad mark and Sonya is able to slow the bad marks down. All these abilities work on cooldown – that means that if you catch a compile error the ability is disabled, and you have to wait for some time.

The other game is our interpretation of the famous DoodleJump, again, with respect to our everyday life in HSE. The functionality is rather simple: the player controls the chosen character with keyboard and jump from one platform to another, trying to stay in the game for as long as he or she can and avoid the obstacles. The game ends as soon as the character falls and the score depends on how far you could go.

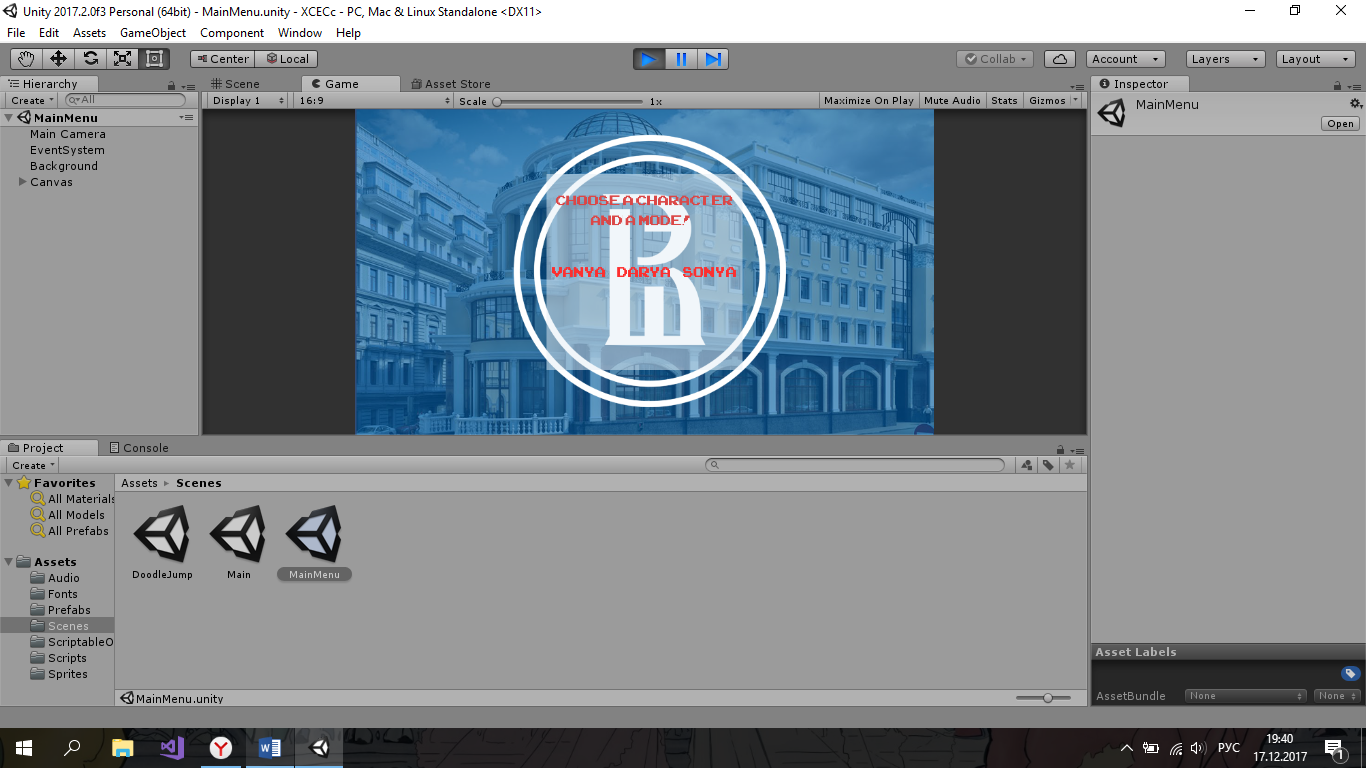
•Central repository address - <https://github.com/SadykovaDarya/HSEss>

•Members of the team and their roles

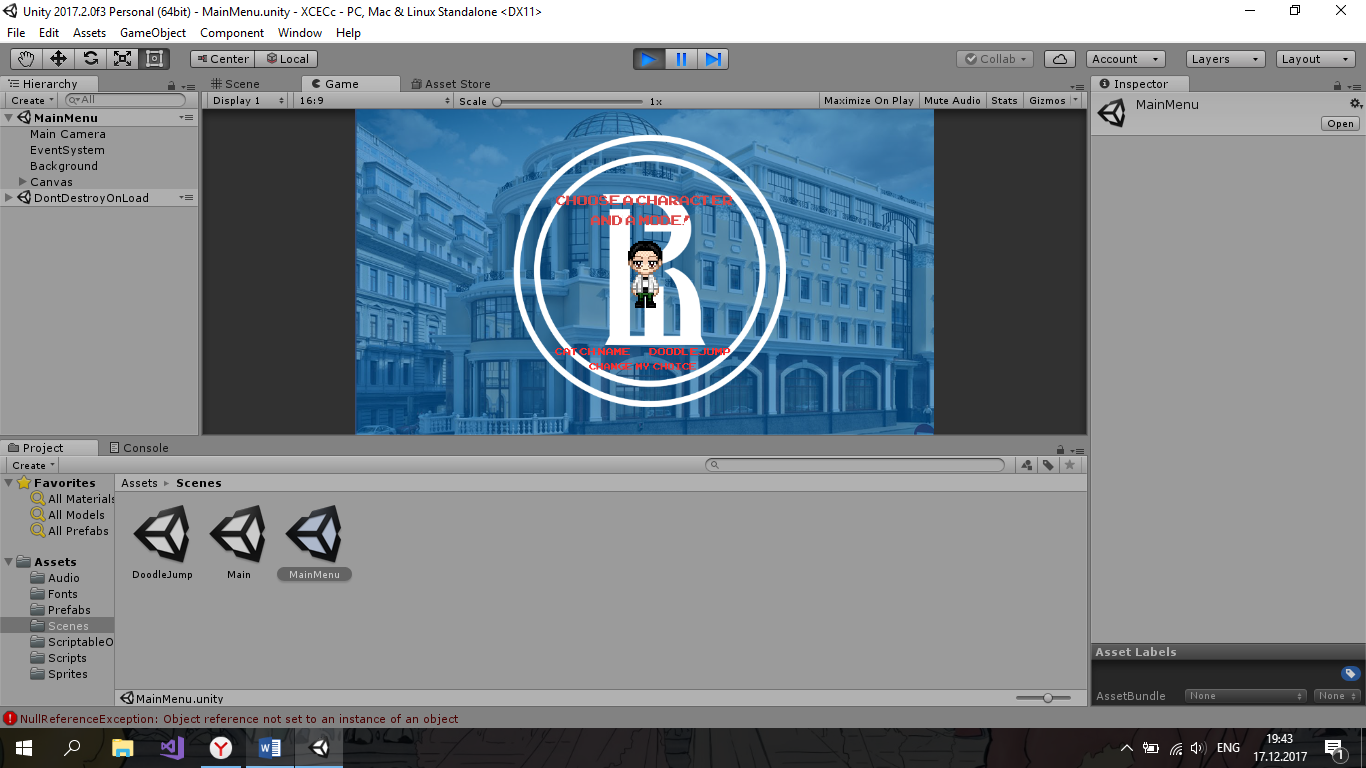
* Tarasova Sophie, BBI161: game logic, unity engine
* Tominets Ivan, BBI162: main logic, unity engine
* Sadykova Darya, BBI161: UI, unity engine

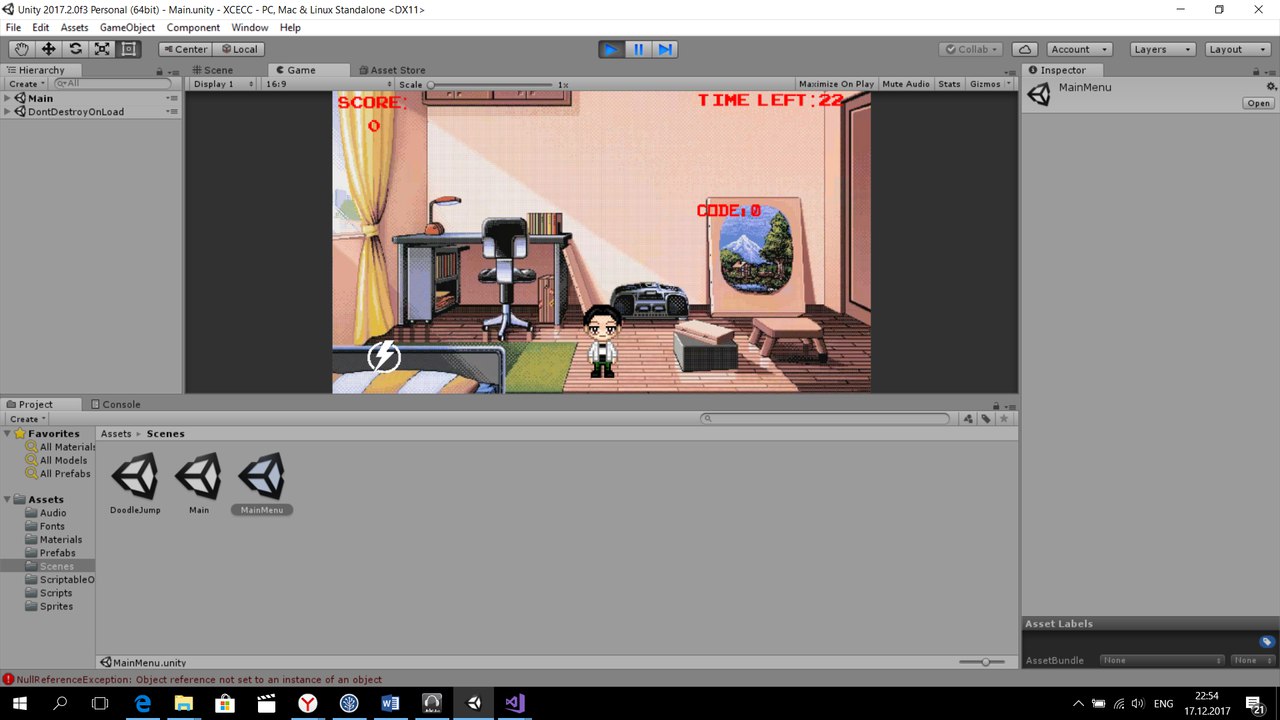
•List of classes with short description for each class (annotation for each class member is not required)

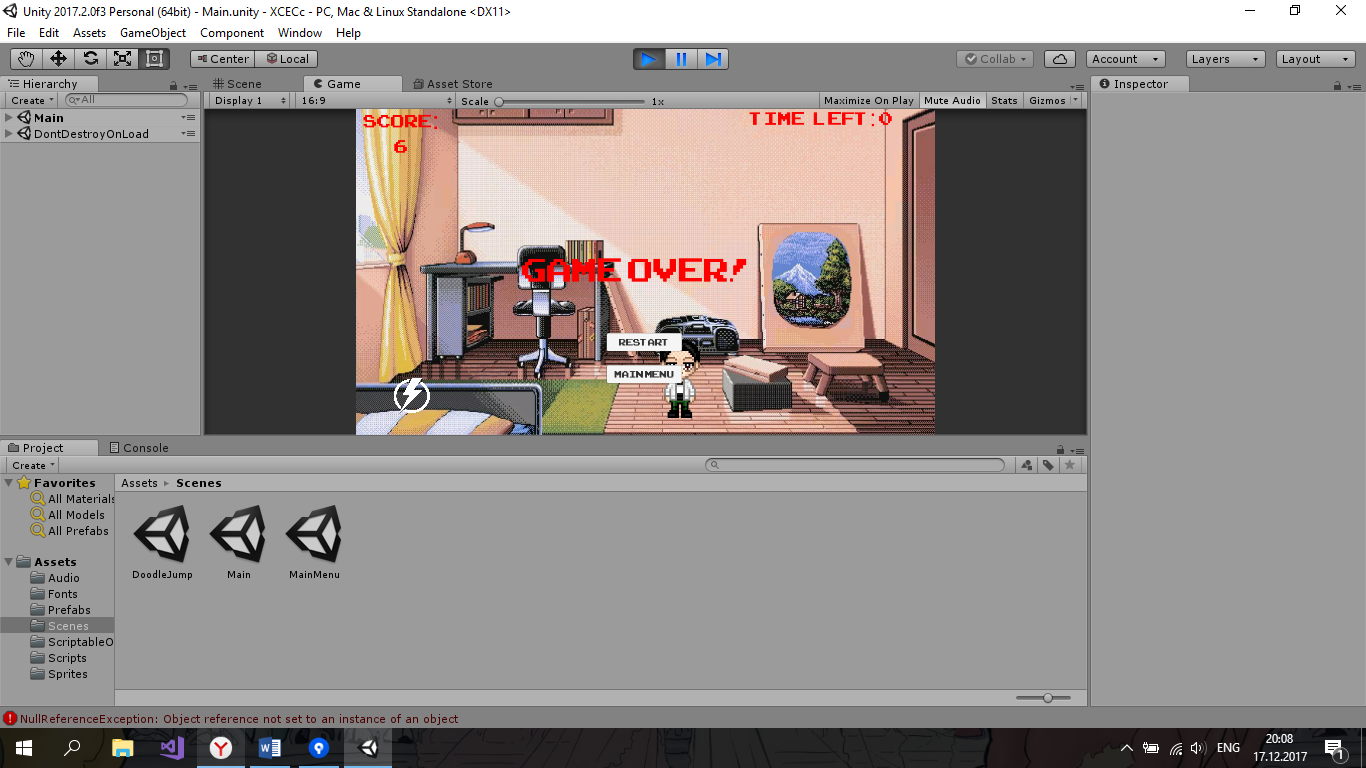
* Ability: this script is almost the same as our usual classes, it contains necessary fields for all Ability scriptable objects: their names, sprites, etc.
* AbilityCoolDown:
* BackgroundBehavior:
* BackToMain: this script is used on all the buttons that allow player to back to main menu and restart the scene. Therefore, there are only two methods. BackToMain destroys the character in the current scene and then navigates us to the menu, Restart does basically the same, but also returns the current scene to the initial state.
* BadMarkBehavior: this script checks whether the collision happened with the right character and then slows the mark down. Also it sets the disturbing sound on collision with every character.
* CameraBehavior
* Character: this script is almost the same as our usual classes, it contains necessary fields for all Character scriptable objects: their names, sprites, etc.
* CharacterSelector: this script is responsible for the selection of the characters. It understands what character was chosen by the index of the clicked button and then spawns the prefab with all the necessary properties. There are also other functions that navigate the player to the chosen scenes and allow to change the selection.
* DAbility: this script is for the Ability of Darya character.
* DontDestroy: this script is assigned to the prefab and allows us to pass it between the scenes.
* DoodleJumpScore: this script is responsible for counting the score, which is basically just the travelled distance, during the DoodleJump scene.
* GameController: this script is responsible for most of the logic in the catch game. It spawns the marks, it controls the timer panel and visibility of the buttons and game over text.
* Jumping
* markBehavior: this script controls the behavior of good marks, their sound on collision and the destroy of them.
* MonsterBehavior: this script controls the behavior of hurdles: rotation, speed, etc.
* Moving: this script controls moving of a character in the DoodleJump scene.
* PlatformBehavior: this script is responsible for controls how platforms behave – enabling a Collider2D component, changing a position when camera goes way up.
* PortalBehavior: this script contains logic responsible for character having an ability to appear on the other edge of a display when going through one.
* SAbility: this script is for the Ability of Sonya character.
* Score: this script is responsible for the scoring system in the catch game and also for checking if the character is Darya: her ability sets special behavior scripting.
* setPlayersPosition: this script is responsible for setting the position of a character in a new scene.
* SpawnMonster: this script controls the spawning of obstacles in the DoodleJump.
* SpawnPlatforms: - this script is used for setting platforms positions on the start of the DoodleJump scene.
* StudentController: this script is responsible for controlling the player in the Main scene so that it moves neat and smooth by the key arrows.
* VAbility: this script is for the Ability of Vanya character.

•Program interface (make screenshots of the key forms/windows and add annotations to them)

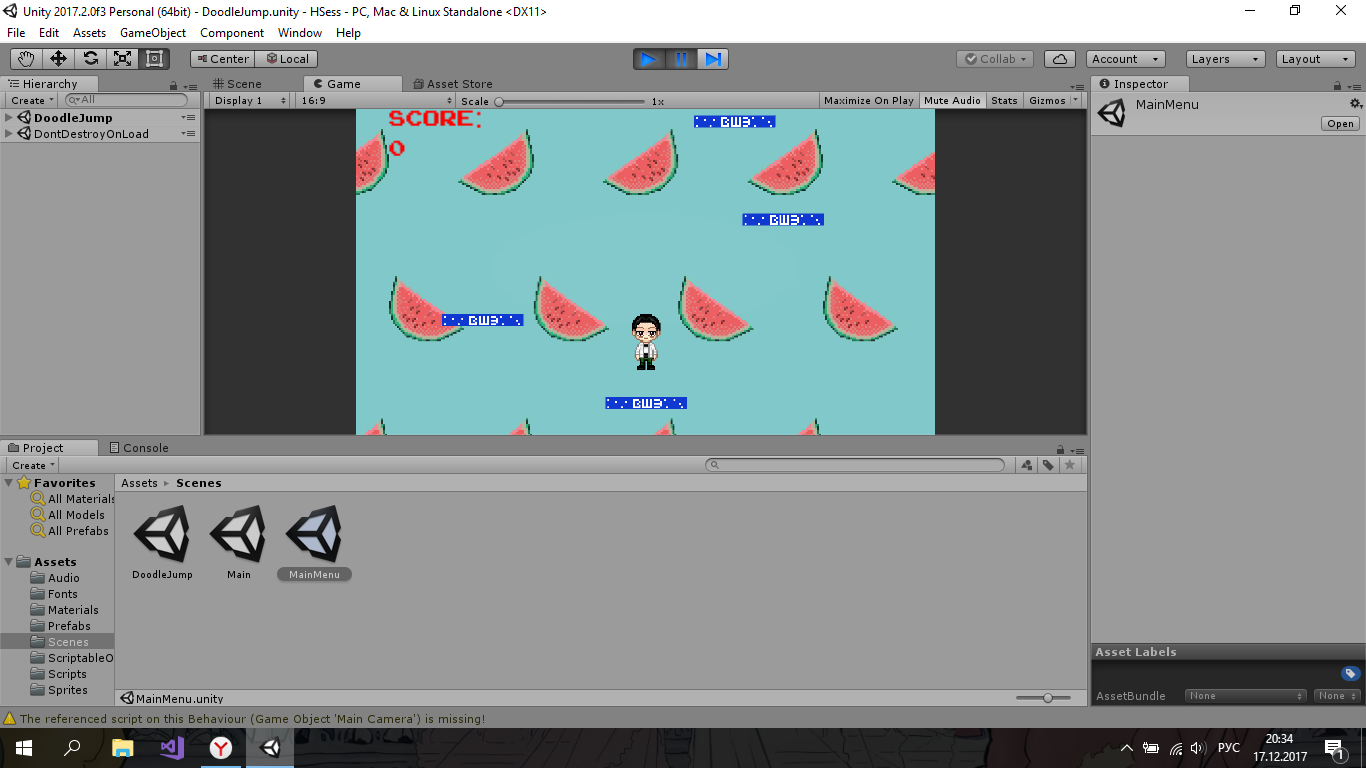
Main menu scene – a scene where you can choose a character and a mode.

The same scene after the choice (we are going to choose Vanya with no particular reason, other characters are also clickable). The player also can change the selection.

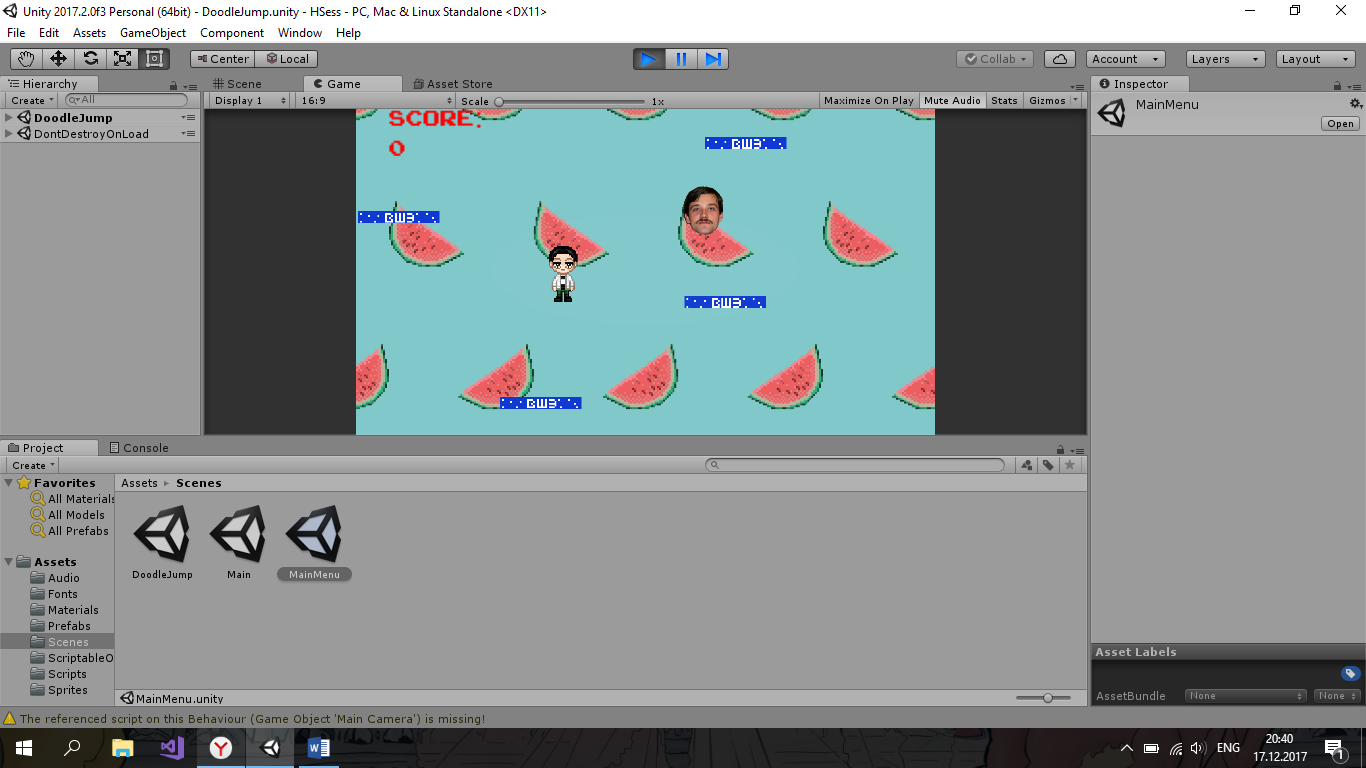
The catch game scene. This is where the gaming itself takes place, catching marks, scoring etc.



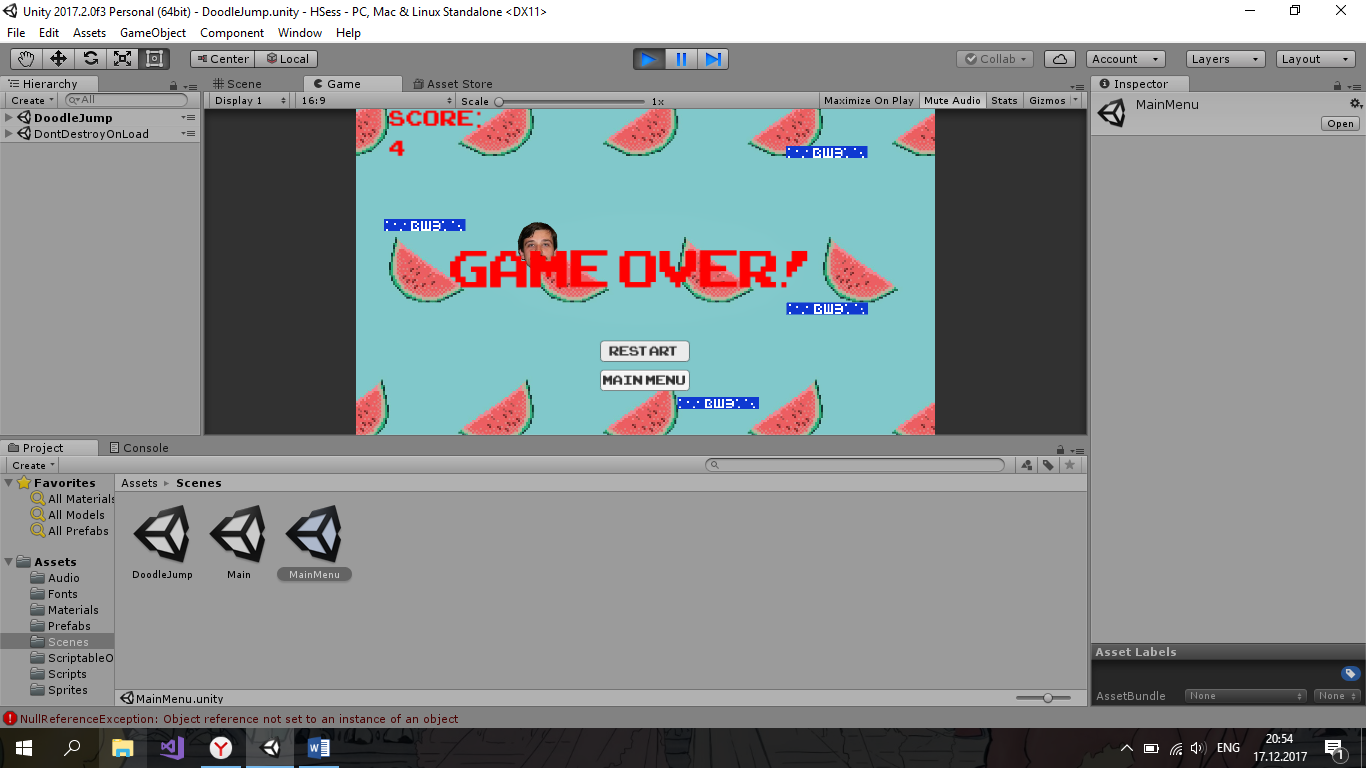
This is the end screen of the catch game, here you have a choice: to restart the game and try again or to go back to the main menu.



This is the scene of DoodleJump, nice and simple. At some point hurdles are going to appear – we’ll get to it later!



And the hurdles are the faces of our calculus teachers! (we are sorry if it seems inappropriate to you, it was made only for the sake of joke).



The game over scene for the DoodleJump, done in the same style as the catch game’s one!

That’s basically it!

•Test cases.

We think that we tested all the scenarios that may cause exception and we are glad to provide them. They include choosing the character, playing with it then rechoosing it again and playing to check if it doesn’t appear in the scene twice; multiple losing in the DoodleJump to check the right spawn of a character; multiple replaying of each scene with all characters to test their abilities; checking the sounds on all the collisions and scenes.