# Astro Game Design Document

1. Title Page
   1. Astro – surviving the planet of unknown
2. Game Overview  
   An astronaut crashed his ship in an unknown planet full of resources and hostile creatures. While crashing, his ship fell into pieces but the main housing was still working. Trying to collect all the missing pieces and find the resource needed for escaping the planet, he will get involved into a lot of trouble and adventures.
3. Genre – 2D Platformer
4. Target Audience – kids, young adults, could fit in every audience
5. Game Flow Summary – the 2D player movement is allowing the main character to move in 3 different directions –

* Jump and do a flip
* move forward (right)
* move backwards (left)

1. Look and Feel – pixel art in cave-like formations
2. Gameplay and Mechanics
   1. Gameplay
      * The character has a main weapon with which he is able to shoot and kill the aliens in the level before they kill him, he also has a time rewind ability in which he can go 5 seconds back in time. This is implemented in the TimeBody.cs and PointInTime.cs scripts. This ability is made possible by his special space suit. The player has a health bar and there are different types of dangers spread out in the level. In the PlayerHUD.cs script the health system and all of the different damage types are implemented. When damaged, the player will flash red to indicate that he has taken damage. The health bar has 3 states – high, medium and low indicated the 3 sets of colors which are green, yellow and red respectavly. This functionality is implemented in the HealthBar.cs script. The player is driven by an animator which contains 3 states in which the player can be and these are: player\_idle in which the main character is not moving and is staying still; player\_run in which the player is moving; player\_jump in which the player is jumping and reaching for the platform. All player sets have their sets of animations.
      * There are 4 types of dangers in the level:
3. Aliens – crab-like enemies and octopuses which do damage to the player if they touch him and have a patrol pattern. Their behavior is implemented in the PatrollingEnemy.cs, PatrollingOctopus.cs and Enemy.cs scripts in the project scripts folder.
4. Turret – a column styled turret which engages the player if he gets in their line of sight. The turret appears from the ground when it is awoken and starts attacking the player and shoots at him until he gets out of its line of sight. The scripts implementing the AI of the turret is in TurretAI.cs. AttackVision.cs and TurretBullet.cs scripts. A whole set of animations are implemented for the turret using states to determine what animations to play.
5. Spikes – spikes are spread across the level and if the player touches or falls on to them he will take damage and will be knocked back.
6. Falling platforms – there are platforms across the level that when touched after a countdown (1 sec) will start falling and if the player is on them he will fall to his death. The falling platforms are implemented in the FallingPlatform.cs script.
   * + Mission/challenge Structure - acquire parts to fix your ship while searching through the world and overcoming its dangers.
     + Objectives – collect you ship parts to leave the planet and survive.
   1. Mechanics
      * Physics – human-like movements and collisions, gravity and force
      * Objects – might be moved or go through them depending on the progression
      * Actions – player has jump, shoot and time rewind ability and interaction with environment
      * Combat – player can shoot with the blaster in his arm, hostile creatures can kill him if they are close enough. Shots from turrets must be avoided because they are invincible to the player.

8. UI

The game has three menus:

1. The main menu from which the player can start playing, has options section for volume control and a quit button to exit the game.

2. The Game Over screen – when the player health falls below zero the player dies and this screen appears informing the player that he is dead. He has the options to start all over again or to go back to the main menu.

3. The Game Won screen – this screen appears when the player has successfully collected the space ship parts and he has 2 options: to play the level again or to exit the game.

Scripts:

PlayerHUD.cs – this scripts controls the health system of the player. It consists of the player health, the different damage types, the player death scenario and the different damage types. The damage type is determined by the gameobject tag with which the player collides and whenever there is damage taken, the damage animation is played to indicate to the player that he is hurt.

PlayerMovement.cs – this handles the animations and movement of the main character with the help of the built in CharacterController2D script. From here it is determined in which state does the player find himself in and which animations to play.

Bullet.cs, PrefabWeapon.cs and RayCastWeapon.cs – these scripts handle the attack logic and behavior of the main character. The projectiles and damage of the player weapon are implemented in Bullet.cs and if a collision between the bullet gameobject and a gameobject of type enemy and hit is detected and part of their health is taken. If the given health falls below zero a death animation for the enemy plays and the enemy is destroyed. PrefabWeapon.cs handles the player input and initiates the attack. In RayCastWeapon.cs the attack vision of the player is implemented which draws a line in the line of site of the player which plays the role of his attack vision. If anything in this vision is hit the bullet is destroyed and if an enemy is hit damage is taken by them and the given animation is played (impact and destruction of an enemy).

PatrollingEnemy.cs, Enemy.cs and PatrollingOctopus.cs – these scripts handle the behavior of the mobile enemies in the game. The health system of the enemies is handled in Enemy.cs as well as their death state. If their health falls below zero, the death effect animation is played and the enemy game object is destroyed. Also the score of the player is incremented by 100 point per killed enemy. The patrol pattern of the crabs and octopus enemies function the same way. A 2D ray is casted using RaycastHit2D from a point attached to the enemy game object and checks for a given layer in the game using the LayerMask component. In our case checks if there is ground below the enemy on which it can walk. If the end of the ground is reached the sprite and the direction of the movement are flipped. Also there are 2 more checks for the spikes and turrets in the game to prevent the enemies of going through them.

TurretAI.cs, TurretBullet.cs, AttackVision.cs – TurretBullet.cs handles only one very simple thing. After a bullet is fired from the turret a check is made if this bullet is colliding with a trigger collider or not. If it is it is destroyed. AttackVision.cs determines in which direction the turret will be firing which is determined by the position to the player relative to the turret. TurretAI.cs is the main script for the turret enemy type. Here is the attack pattern of the turret implemented as well as it’s animation states.The attack range and firing of projectiles is implemented.

Platform and FallingPlatform scripts – the two special platforms are implemented here – the moving ones and the falling ones. The moving platform function in a very simple and elegant way. It has two end points attached to it. The left and right end point and it is checked if the platform has reached any of them and its position is updated. The falling platform functions as a trap and if the player falls on it a counter starts which is set to 1sec. After that the kinematic property of the platform game object is disabled and the platform falls in the scene.

TimeBody.cs – this script implements the time rewind ability of the main character. It stores the past positions of the player before the button ability is pressed and can restore his past position up to 5 sec back. Also a grayed out effect of the screen is played to indicate that the ability is in use.

Developers

Георги Енев, F97402 – moving platforms implementation and rewind ability of main character.

Роберто Божичков , F87300 – UI and ship parts collection mechanic and score system.

Мартин Калчев, F85994 – level design, obstacles and enemy animations and implementation , health system and character control and animations.