Endlessrunner Konzept

Nr	Criterion	Explanation	Implementation
0	Units and Positions	Where is 0, what is 1? Explain your setup of coordinate systems of the entities.	The endless runner is focused on the character. This means the camera follows the x-translation of the character (but doesn't follow the Y-Translation (when jumping for example) The character is 1.5m high and 0.5m in width. The Game is visualized in 2.5D (see Figure 1)
1	Hierarchy	Explain the setup of the graphs and the advantages you gain from them.	- Consists of 3 child nodes of scene graph (PlayerAgent, Terrain, NPCs) - PlayerAgent contains all Scripts and other parts of the player agent and will likely have children who contain bones and other components - Terrain contains the different parts of the terrain (like platforms, obstacles, possibly items) and their respective scripts. Elements will also be dynamically added to the terrain (before entering the player screen)
2	Editor	Use the visual editor and explain which parts are better done by coding and why.	I will mainly use the editor to build the different elements (player agent, terrain elements, and NPCs) and generate prefabs. I will also use it to build the initial scene graph (containing a first part of the level to which then later elements are being added)
3	Script components	Use script components and explain if they were useful in your context or not and why.	I want to use the script components for organizing the different behaviors - Scripts for NPCs - Scripts for UI elements - Scripts for Collectables
4	Extend	Derive classes from FudgeCore and explain if that was useful in your context or not and why.	- PlayerAgent script derives from Node class as it will likely contain all the behavior of the player agent and through that can easily be instantiated at the beginning based on external data
5	Sound	Use sounds and explain your choice of sounds and placement concerning the user's perception.	Sounds for: - PlayerAgent (walking, hitting ground/NPC/Item, dying) - NPCs (movement, maybe dangerous noises, dying) - World (Background music)
6	VUI	Create a virtual user interface using the interface controller and mutables. Explain the interface.	- Score UI element - a menu for adjusting settings (volume, difficulty) - health bar - death screen (contains score and playing time and start new button)
7	Event-System	Use the event system to send messages through	Events for communication between game elements, e.g., NPCs fire event when dying,

		graphs and explain if that was useful in your context or not and why.	lead to an increase in score and maybe a slow increase in difficulty
8	External Data	Create a configuration file your application loads and adjusts to the content. Explain your choice of parameters.	Allows modification of settings like difficulty, master volume, the movement speed of camera/character, and occurring NPC types (e.g., dragon yes, turtle no)
9	Light	Explain your choice of lights in your graphs.	Since it is 2.5D I want to use light for highlighting elements, like items or NPCs. Accordingly, the background light is put on the graph while the respective child nodes (NPC or Item)
A	Physics	Add rigid body components and work with collisions (1) and/or forces and torques (1) and/or joints (1)	Using rigid bodies for collisions between player and ground elements but also with NPCs and Items, applying impulses to the character for realistic jumping, and using trigger elements e.g., for Items Using ray casts for NPC behavior
B	Net	Add multiplayer functionality via a network (3)	--
C	State Machines	Create autonomous entities using the StateMachine (1) and/or ComponentStateMachine (1) defined in FudgeAid	For NPC behavior (patrol, attack player, die, killed player) and game state (main menu, playing, death screen)
D	Animations	Animate using the animation system of FudgeCore (1) and/or Sprites (1) as defined in FudgeAid	-

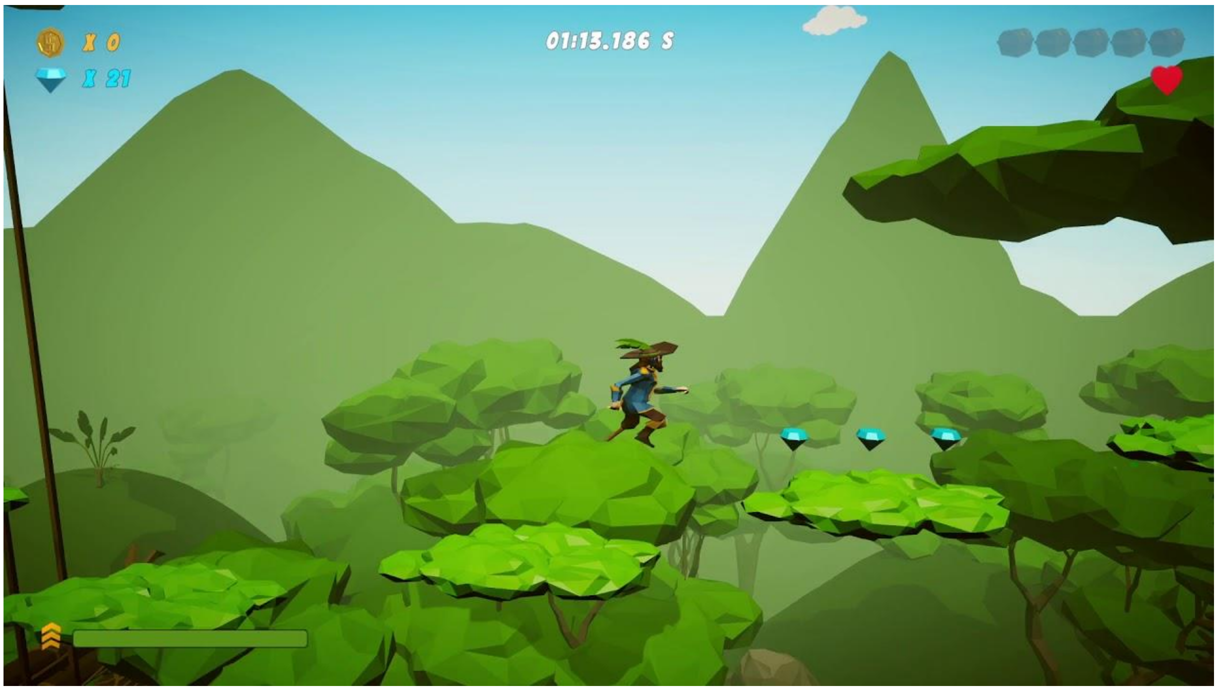


Figure 2: 2.5D in Jump and Run

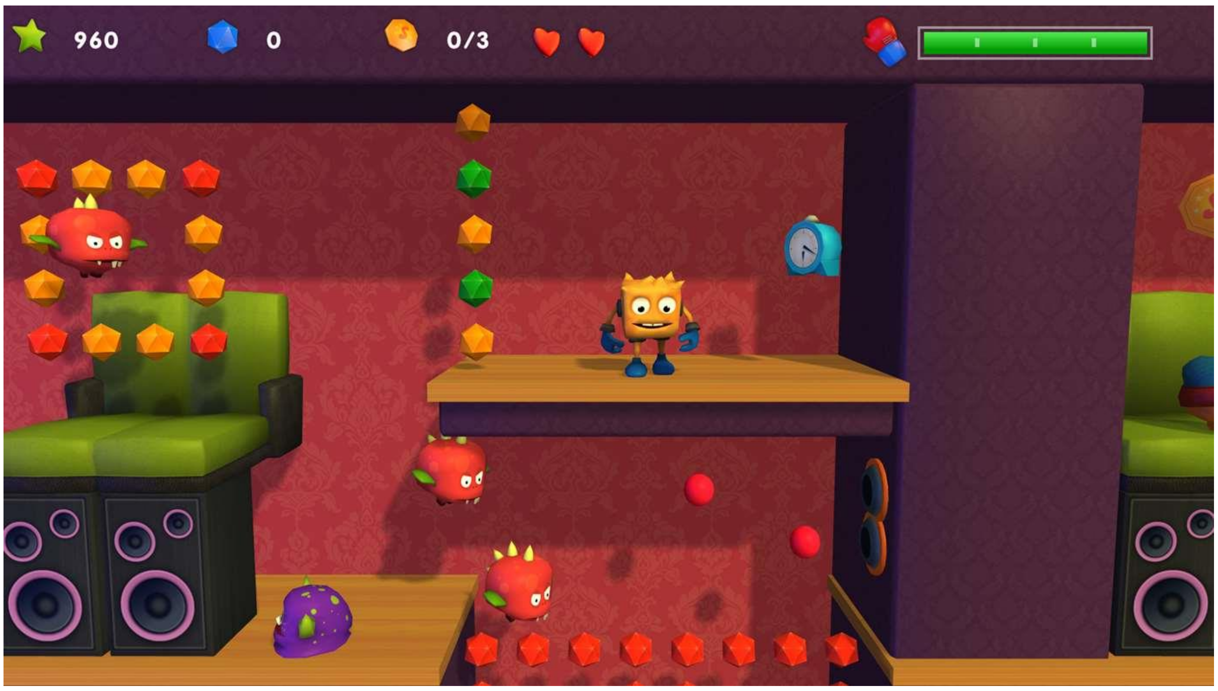


Figure 1: 2.5D with enemies and collectables

Possible Hierarchy:

SceneGraph

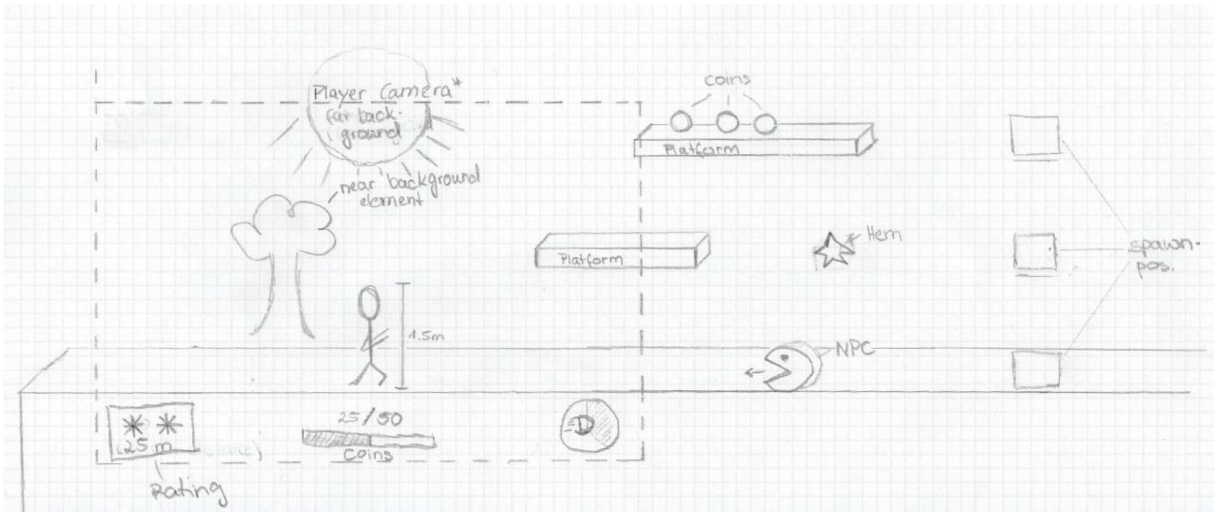
- PlayerAgent
  - PlayerBody
- Terrain
  - Ground
  - Platform(1)
    - Coin(1)
  - Platform(2)
- NPCs
  - NPC(1)
  - NPC(2)

Game Mechanics:

- Collect coins while playing
  - More coins mean faster progress (1 coin = 10 points = 10 secs worth of playing)
- Ground is endless (will be prolonged whenever the player gets close to the edge; before the edge is visible on screen)
- Terrain elements (platforms) are procedurally and randomly generated at spawn positions outside of the screen
- Coins, NPCs & obstacles are placed on empty free spaces between and on terrain elements
- NPCs: flying enemy that waits for a few seconds and then shoots at the player
- Player dies when running into an NPC or obstacle

Interactions: Spacebar – Jump & Drop

- Jump can only be executed when grounded, or one-time mid-air
- Drop can only be executed when mid-air



- Player Camera FOV larger than in sketch

- Note: Removed Items and background elements from previous design concept