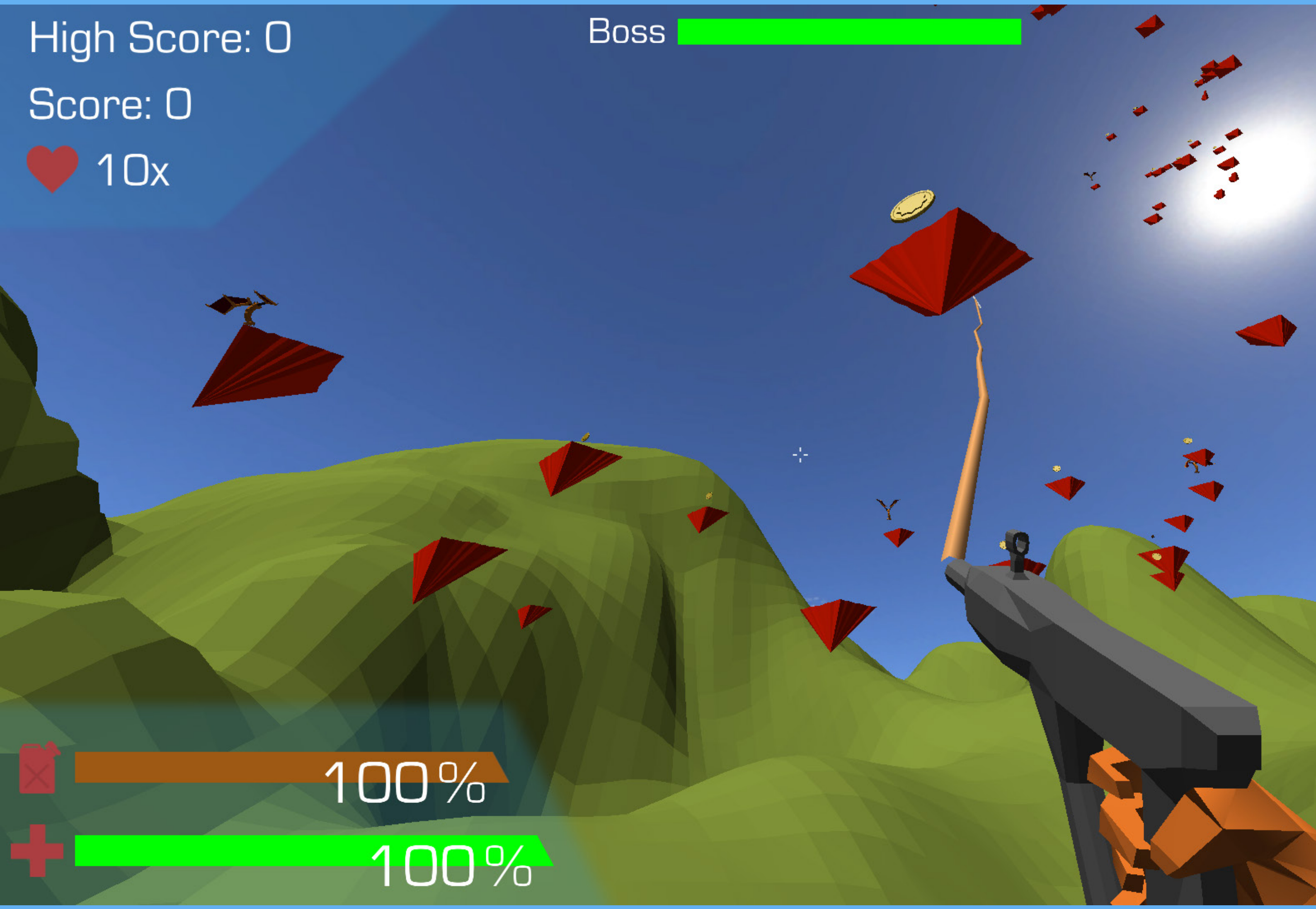
INFINITY



Created By:
David Akkerman
Dorus van den Oord
Lieuwe Locht
Erik Veldhuis
Daan Picavet

unity

TABLE OF CONTENTS

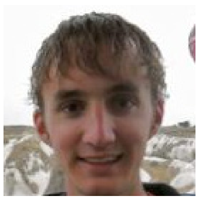
INTRODUCTION

In this report our game, Infinity, made during the minor “Software Ontwerpen en Toepassen” and the process during the making is discussed. In the first weeks a theme was chosen to define the game’s content and gameplay. The chosen theme was “You only have one tool”. The initial idea was to build a game with procedurally generated levels in which the player has one tool to move him through the level and tackle problems along the way. If the end is reached the player is teleported to a boss level. If he succeeds, the player loses his tool and gains a new one.

We managed to build two different procedurally generated environments, quite an arsenal of tools and three boss levels. The next pages will guide you through the process and it’s final product.

TEAM

Producer: David Akkerman, TN, Faculty of TNW, 4220390


Lead Artist: Dorus van den Oord, IO, Faculty of IDE, 4215567

Lead Programmer: Erik Veldhuis, WB, Faculty of 3ME, 4117425

World Builder: Lieuwe Locht, TN, Faculty of TNW, 4229681

Game Designer: Daan Picavet, IO, Faculty of IDE, 4154517

TARGET AUDIENCE

The target audience for this game are people whose ages range from 8-40 years. It is too difficult for children who are younger than 8 years old and it is not interesting for most people over 40. The game should appeal to anyone familiar with an FPS, but the game should also be attractive to players who are new to the genre.

PLATFORM & CONTROLS

The game is to be released on Windows PC

GENERAL	Walking	W A S D
	Looking	Mouse
	Jumping	Spacebar
	Pause Menu	Escape
	Equip Tool	1
GRAPPLING HOOK	shoot	Left Click
	detach	Right Click
	extend rope	Q
	retract rope	E
POGOSTICK	Jump (Higher)	Spacebar
	equip/uneqip	Left Click
BLACK HOLE GUN	Charge	Left Click Hold
	Shoot	Left Click Release
	Destroy	Right Click
HANDGUN & JETPACK	Toggle Handgun	1
	Toggle Jetpack	2
	Use Jetpack	Spacebar
	Shoot	Left Click
	Reload	Right Click
SMG & JETPACK	Toggle Gun	1
	Toggle Jetpack	2
	Use Jetpack	Spacebar
	Aim	Left Shift
	Shoot	Left Click
	Reload	Right Click
SNIPER & JETPACK	Toggle Gun	1
	Toggle Jetpack	2
	Use Jetpack	Spacebar
	Aim	Left Shift
	Shoot	Left Click
	Reload	Right Click
	Zoom	Scroll Wheel

COMPONENTS

These are the components implemented in the game. This is an updated list of the list in the previous report, since there were a few features implemented in the game, not appearing on the list.

Component	Stars	Implementation	Responsibility
3d Models - CG	*	3D Animated Environment	Dorus
3d Animated Models - CG	***	attacking, walking, standing	Dorus
Proc. Generated Mesh - CG	***	Levels procedurally generated	Lieuwe
Animated Textures - CG	**	Lava texture	Erik
SoundFX - CG	*	Noise, Attack and Player Sounds	Daan
Soundtracks - CG	**	Compose 3 in-game, 1 menu	Daan
Camera Shakes - CG	*	While Shooting (recoil)	Daan
Unsteady Camera - CG	**	Walking/Jumping	Daan
Particle Systems - CG	*	Explosions/Pickups	Dorus
Start/Pause/End Screen - CG	*	GUI	Dorus
Highscores - CG	*	Killing enemies/Progress	Erik
Options - CG	**	SoundToggle/Resolution etc.	Dorus
Credits- CG	*	Credits to us	Dorus
Pathfinding Own Algorithm - AI	***	Enemies tracing the player	Erik
Consciousness - AI	***	Enemies 'notice' the player	Erik
Learning Enemies - AI	***	Enemies learn while fighting	Erik
Data Trough Playthrough - WD	**	Achievements/Progress	David
Data On Server - WD	**	Account + Achievements	David
Visualize data on server - WD	**	Achievements/Skills	David
Show Score From Server - WD	**	Save Highscores to cloud	David
Proc. Generated Levels - PR	***	Levels are generated during play	Lieuwe
Moving Platforms - PR	*	Prefab of moving platform	Lieuwe
Time Independent - PR	**	using Time.deltaTime	Daan
Custom Physics - PR	**	Tool that alters speed of NPC	Erik
Multiple Cameras - PR	*	Weapon camera,Zooming Sniper	Dorus
Unity Triggers - PR	**	collision/force/detection	Erik
Multiple Weapons - PR	***	Different Weapons	Daan
Multiple Enemies - PR	**	Bosses and regular enemies	Daan

All features are partially to fully implemented.

CG - 21 ★

AI - 9 ★

WD - 8 ★

PR - 16 ★

3D Models & Animations

The models are made using Blender. The first model is the grappling hook. After that we decided to animate the tools with arms in a first person way. This armature is done by Erik. All tools and animations are done by Daan, except for the Pogo stick (done by Erik). Erik has done the environment and model of the pogo stick boss and also made the Skybox. Daan did the remaining boss models & animations. He also made the models & animations of the smaller enemies.

Procedurally Generated Meshes

These are incorporated in the procedurally generated level script from the underground environment. This is done by generating a 3dimensional matrix with a binary value: accessible or inaccessible terrain. Between these areas a wall of triangles is generated. This was created by Lieuwe. It took a lot of effort to get the system to work properly, because the triangles had to connect properly.

Animated Textures

An animated texture was made by Erik to impersonate the lava.

SoundFX

The sound effects were done by Daan in FL Studio using synths and modifying samples. Environmental effects are made to give the game a better feel.

Soundtrack

The Soundtracks was also made in FL Studio by Daan. The goal was to give the player a more explorative experience.

Camera Shakes

These effects are added when firing one of the guns. This was done by Dorus to give the guns a realistic way of having recoil. Also Erik implemented a camera shake when the player is hit.

Unsteady Cam

When the player is walking the camera's y-position is transformed by a sinusoid with an amplitude depending on the speed of the player. This is done by Erik.

Particle Systems

Particles are used to simulate explosions when enemies die and the portal effect. These are done by Daan. Also a few other Particle Systems are used in Pickups and the Pogo Stick Boss, by Erik.

Start/Pause/End Screen

These are made using the Unity UI. The Pause screen contains a lot of nested elements to make the options and other available during the game. These are made by Dorus. This took a long time, because there is little documentation on the internet and the UI is counter-intuitive.

Highscores

The in-game highscore is saved in the Playerprefs. Every pickup and Enemy has it's own score value. This is done by David.

Options

Options are created by Dorus using the Unity GUI system. Options are made using the Screen-class and AudioListener-class.

Credits

The credits are made by Lieuwe using plain text.

Pathfinding

The helicopter- and bunny-enemy make use of 5 sensors (ray-casts) spread over the 180 degrees in front of the enemy. The sensors detect the space between the enemy and a wall. The fitness is set by the enemy that gets closest to the player.

Group detection (Consciousness)

Particle Swarm Optimization is used containing a random factor, so the enemy will not be stuck in a local optimum. The rotation is updated using this CI method.

Hazard detection (Consciousness)

Sensors pointing downwards, checking for lava or deep holes. If so the coordinates are saved and the enemy will avoid this point.

Learning Enemies

If the pathfinding sensors detect a wall, it counts the number of detection and it will increase or decrease the sensor's height or length. All AI parts are doen by Erik

Data Through Play Through

The game keeps a record of how many enemies you've killed and beating certain levels gives you achievements.

Data On Server

Data is saved on your laptop and will also be sent and saved in the SQL database online using your username and password.

Visualize Data & Show Score From server

The site displays highscores of all players and your own achievements. The Web &Database part is done by David.

Procedurally Generated Levels

This done by Lieuwe. A chunk system is used that works like the one in minecraft.

Moving Platforms

A Prefab is made to be placed at the beginning of a game level.

Time Independency

Every timed action or counter in the gameplay itself is done using Time.deltaTime.

Custom Physics

A Gravity Gun is programmed by Dorus to shoot a ball attracting and reflecting all physics objects near it. The Grappling Hook has custom Physics done by Erik.

Multiple Cameras

3 cameras are used: 1 for the player's FPS, one especially for his tool and one for the sniper zoom. These are made by Dorus. It was difficult to get all the elements like the skybox in there .

Unity Triggers

A lot of Collision triggers are used to detect whether an enemy was shot. Everything with a rigidBody-component can be targeted by bullets or the grappling hook.

Multiple Weapons

The Pogo Stick and the grappling hook, including rope physics, is programmed by Erik. The other tools are done by Dorus.

Multiple Enemies

Enemies are programmed by Erik and David. Gravityboss is done by David, Pogostick Boss is done by Erik and GrapplingHookBoss is done by Daan.

CODE QUALITY

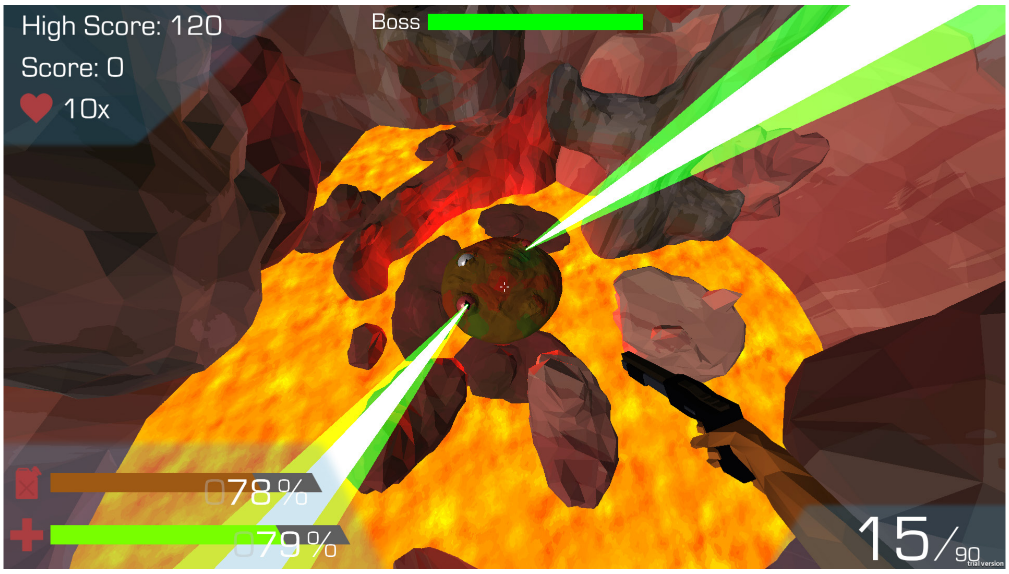
There were a lot of tasks easily to be done by one person. But for instance all tools needed to interact with enemies and the environment. Often someone would write scripts and make assets on their own PC and export them with an explanation in the commit messagebox. When working on a beta or final version, there was always someone in charge of the project and working with everyone separately to implement all asssets. Also there are comments to explain the major goal of a script.

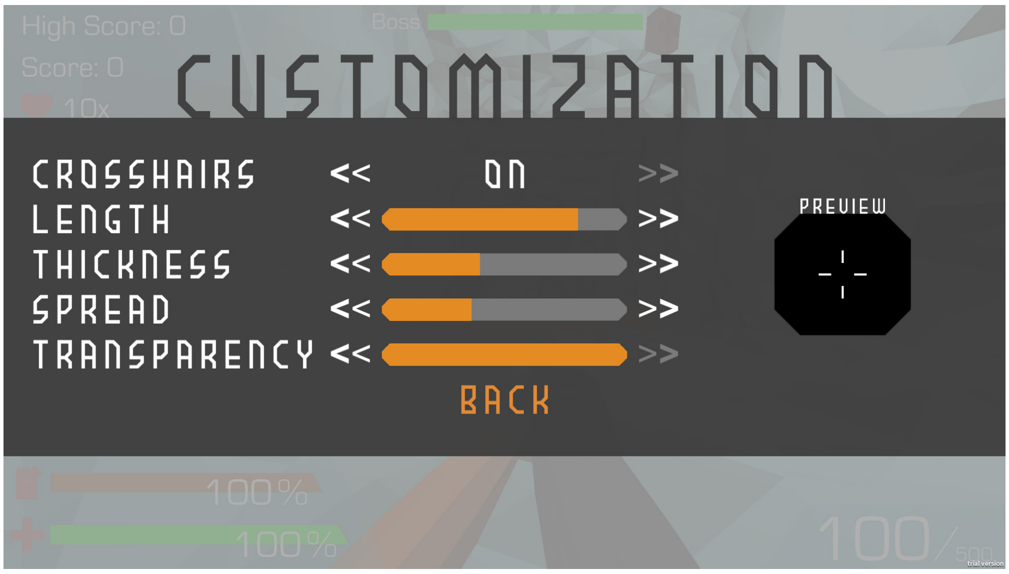
SETTING

The setting is a 3D world that has an explorative character, where the player has to find a way to reach the exit and be inventive to overcome obstacles along the way. When played, it has the feeling of being in a huge world without an ending, never knowing what to find in the next cavern or passageway.

ART

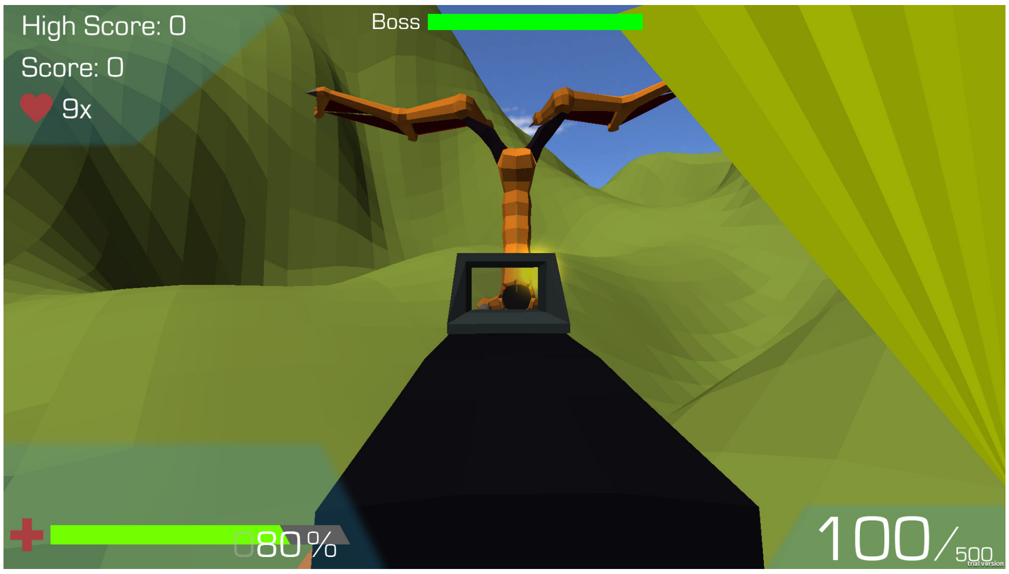
The main art-style we chose at the beginning of the project was 'Low-Poly'. This because it was an easy style to maintain throughout the project and models could be made quickly without the game looking incoherent, models are done by Erik and Daan. In the main menu a triangle-like font was chosen to fit the style. The in-game GUI is made to resemble a heads-up display with the directly important information on the lower end and the less directly important on the upper end of the screen. All of the UI is resolution- and aspect-ratio-independent.





CUSTOMIZATION

CROSSHAIRS << ON >>
LENGTH << >>
THICKNESS << >>
SPREAD << >>
TRANSPARENCY << >>
PREVIEW
BACK



PROCESS

At the start of each week all issues on GitHub were discussed and evaluated. Small tasks were completed each week and more difficult tasks were spread over multiple weeks. Also new tasks were devised and assigned to students who did not have any tasks left. In the end the issues on GitHub were used less, because everyone was finishing up their tasks and just working on several things at once to get the release as complete as possible. For instance debugging the game to make it ready for a release.

During the project the roles were merely a way to make sure everyone was responsible for the implementation of the components, so they would all be included in the game. During the project everyone made sure their responsibilities were

CONCLUSION

Upon release, the game was working pretty fine. There are more tools and enemies implemented in the game then we at first thought was possible in the given amount of time. The game has a really consistent look, because everything is kept in low-poly style.

A few things that went really well were the creation of weapons, enemies and bosses and animating them. Everyone was motivated kept their promises and finished their issues in time (mostly).

After gaining knowledge of how Unity works and producing the game, the time and effort required to finish issues can be estimated more accurately. Also Unity can be used in a better organised way. Next time there can also be made more use of tags, because using public variables in combination with the inspector breaks sometimes when exporting as a unitypackage.