

**Computer Science Bachelor’s Degree**

Dep. of Mathematics and Computer Science

**Virtual Reality Course**

**PowerCraft GDD**

**Summary**: *PowerCraft* is a videogame developed in Unity/C# for the Virtual Reality 21/22 course. It is heavily inspired by Minecraft, of which it might be considered a clone. The game diverges from the original on the progression & goal perspective: unlock and utilize electricity. It is generated by *generators*, used by *machines*.

**Professor**: Carmelo Macrì **Student**: Serafino Giovanni Bueti

# Genre

The genres in which *PowerCraft* fits best are reported below.

* Open world
* Survival
* Adventure
* Sandbox

# Game Mechanics

Being open world, the player can freely explore the surrounding areas. Exploration happens *by feet*: the player can **walk**, **run**, **jump**.

Exploration drains the player’s **Trion**: walking is the least expensive, running has an average cost and jumping ranks highest.

Trion represents both the player’s **health** and **stamina** – they are combined. Trion is **regenerated** at a constant rate.

The world is made of **blocks**. Blocks have different **properties** and **characteristics** that uniquely determine player-interaction.

Said interaction occurs by **placing** a block, **destroying** one or directly **interact** with one. Destroying a block may require **tools**.

The player can **craft** new blocks, tools & weapons.

**Enemies** spawn from **portals** that randomly appear around the player.

The player has an inventory (and hotbar) of items, made of 27 and 9 slots.

The player has a set of **inventories** at his/her disposal. A block may retain items which may be accessed, upon interaction, through an inventory GUI.

**Combat** takes place using *swords*.

The player can build **generators** that generate **RF** (redstone flux) and **machines** that draw RF from generators to generate work.

# Platforms

Initially PC-exclusive (Windows).

# Goal

At the initial game’s spawn is located a structure that accepts 4 generators. When placed, the generators will be taken as a tribute to *Feed* to the *Beast* and the game concludes.

# External influences

* Minecraft!

# Project description

Taking heavy inspiration from Minecraft, the player spawns in a (in the future procedurally generated!) world and has to grind & fight for survival.

Thrown into this unknown, blocky and *720p* world filled with trees to punch & enemies to kill, a *Beast* demands for power. As in actual electricity, that is. You are to take upon the challenge of *lone Industrial Revolution*.

# Game mechanics

## 7.1 Player

The player has the following characteristics:

* can move around,
* can run,
* can jump,
* can break blocks,
* can place blocks,
* can interact with interactable blocks,
* can collect items from the ground by standing close enough to them,
* can drop items on the ground,
* can craft items.

## 7.2 Inventory

The inventory system is composed of an items hotbar that is always displayed and a larger items storage accessible by pressing the associated hotkey (*E* by default). The hotbar is always populated first when picking items off the ground, the larger storage only as the hotbar is full.

Items can be swapped, moved, distributed and arranged at will.

**A few technicalities regarding the inventory system** (deemed worthy of note as its design is architecturally complex)**:**

An inventory of items is such that at least one *ItemContainer* is its child. *ItemContainer* is a *Monobehaviour* component attached to a grid that gives its items the ability to be picked up, moved around, stacked etc. It is factually possible to access any item through the *ItemContainers* static class. Upon instantiation, an *ItemContainer* registers itself to the *ItemContainers* class by ID (UUID) – registration creates a *Dictionary* *entry*; said dictionary is accessible publicly by reference.

## 7.3 Crafting System

The crafting system allows the player to create new items starting from a set of existing items using either a *Crafting Table* or the *Player Inventory*, which allows the player to craft simple recipes that require at most a 2x2 crafting grid.

The *Crafting Table* is an interactable block that shows a 3x3 crafting grid GUI to the player upon interaction and makes possible crafting more complex recipes.

**A technical note** about recipes:

Each recipe is encoded in a matrix of *itemName*’s. To make it possible for the player to craft a specific item anywhere on the crafting grid albeit respecting the original recipe’s shape, the recipe’s input items matrix is “normalized”: empty rows and columns are removed by shifting the matrix to the top and/or to the left.

Recipes are stored and registered in a *RecipeRegistry* and to allow efficient (O(1)!) recipe recognition a *Dictionary* with a custom *EqualityComparator* (& hashing algorithm) is used.

## 7.4 Blocks

**Definition** (block): the base entity, represented as a cube, that characterizes the world around the player.

**Definition** (stack): a non-unary set of identical blocks that occupy a single inventory slot; it has a maximum cardinality of **64**.

Blocks can have the following characteristics:

* interactable: by pressing the *RMB*, the player can activate the unique block interaction.
* consumable: by pressing the *RMB*, the player can consume the block and activate its effects.
* placeable: by pressing the *RMB*, the player can place the block in the world.
* breakable: whether the block can be broken.
* smeltable: whether it can be smelted/cooked in a furnace.
* mining level: hardness of the material of which the block is made of – determines the minimum level of hardness required for an item to be able to break it.
* drops: list of blocks & items that can be dropped when broken.
* burn time: the amount of time the item burns in a furnace when used as a fuel source.

## 7.5 List of items

**Wood**

Non-interactable, placeable, crafting. Default damage and mining level. Dropped by punching trees. Burn time: 4s.

**Wooden Planks**

Non-interactable, placeable, crafting. Default damage and mining level.

Crafted from *Wood*, dropped by breaking *Wooden Planks*. Burn time: 1s.

**Leaves**

Non-interactable, placeable, breakable. Default damage and mining level.

Can only be found in the world. Drops *apples* (10%), *sapling* (25%). When no neighbouring *wood* is connected, spontaneously break at a random time.

**Cobblestone**

Non-interactable, placeable, crafting. Default damage and mining level.

Dropped by breaking *cobblestone* with any *pickaxe*.

**Grass**

Non-interactable, placeable. Default damage and mining level.

Can only be found in the world.

**Dirt**

Non-interactable, placeable. Default damage and mining level.

Dropped by breaking *grass*.

**Coal Ore**

Non-interactable, placeable, breakable. Default damage and mining level.

Can only be found in the world. When broken with a *pickaxe* of any level, drops a random number of *coal*.

**Coal**

Non-interactable, crafting. Default damage and mining level. Burn time: 16s.

**Charcoal**

Non-interactable, crafting. Default damage and mining level. Burn time: 16s.

**Iron Ore**

Non-interactable, placeable, smeltable. Default damage and mining level.

Can only be found in the world. Mining level: 1. Produces 1 *iron ingot* when smelted.

**Iron Ingot**

Non-interactable, placeable, crafting. Default damage and mining level.

**Torch**

Non-solid block (player goes straight through). Light-emitting block to illuminate dark places.

# Footnotes

A few more technical notes on **procedural**, seed-based, infinite world generation.

Unity allows the developer to procedurally construct 3D models **at runtime**. It is extremely powerful and given the simplicity of the models representing the world (cubes), it was straightforward enough to implement an algorithm that generates an entire chunk mesh of 16\*256\*16 blocks in less than 10ms using vertices, triangles and manual UV mapping: UVs are used in combination with a procedurally stitched texture – at start-up – that is used on chunks to display, for each block, a different texture on different sides of the singular cube.

## 8.1 A personal note

Working on this project was extremely fun; definitely an interesting ride. I grew up and learned a lot of new exciting things which I intend to further cultivate and study (will probably get into 3D modelling & shader programming in the future!). Hopeful this project can be useful to anybody new to Unity who just started out and is willing to learn and have fun.

Thank you!