

开发一个 视频游戏 虚幻引擎 4



多媒体工程学位

学位项目结束

作者：

Jose Maria Egea 运河

监护人：

米格尔·安吉尔·洛萨诺·奥尔特加

2015 年 9 月

理由和目标

鉴于在多媒体工程学位期间学到的技能,使我们能够

涵盖多媒体内容的各个方面,旨在开发一个

测试这些功能的 3D 视频游戏。

对于视频游戏的构建,打算使用Unreal Engine 4引擎。另外,

该项目包括视频游戏设计文档 (GDD) 和详细说明

相同的。

游戏将使用 Unreal 的 Blueprints 可视化脚本系统制作。

引擎 4。此外,它将与 Oculus Rift DK1 外围设备兼容。

对于游戏场景和菜单的构建,将涵盖编辑器主题。

地形、粒子效果、照明、声音等。

目录索引

目录索引	1
1. 简介.....	9
2. 理论框架或现有技术.....	10
2.1.视频游戏概念	10
2.2.制作视频游戏的引擎.....	10
2.2.1.上下文中的引擎	10
2.2.2.市场上的视频游戏引擎比较	11
2.2.2.1.虚幻引擎 4.....	13
2.2.2.2.统一。	13
2.2.2.3 CryEngine.....	14
2.2.3.发动机的选择.....	十五
2.2.4.虚幻引擎 4 架构概念及其蓝图脚本编程系统.....	十五
2.2.4.1。 UObjects 和 Actor	16
2.2.4.2.游戏性和框架类	16
2.2.4.3.代表世界上的玩家、朋友和敌人.....	16
2.2.4.3.1.典当.....	16
2.2.4.3.2.特点.....	17
2.2.4.4.通过玩家输入或人工智能控制 Pawn。 17	
2.2.4.4.1 控制器.....	17
2.2.4.4.2.播放器控制器	17
2.2.4.4.3 人工智能控制器.....	17
2.2.4.5.显示玩家信息	17
2.2.4.5.1.平视显示器	17
2.2.4.5.2.相机.....	17
2.2.4.6.建立后续行动和游戏规则	17
2.2.4.6.1.游戏模式	17
2.2.4.6.2.游戏状态.....	18
2.2.4.6.3.玩家状态	18
2.2.4.7.框架类的关系.....	18

2.2.4.8.蓝图可视化脚本.....	19
2.2.4.8.1.总体蓝图.....	19
2.2.4.8.2.蓝图的类型.....	20
2.2.4.8.2.1.类蓝图.....	20
2.2.4.8.2.2.关卡蓝图	20
2.2.4.8.2.3.蓝图界面	20
2.2.4.8.3.蓝图剖析.....	二十一
2.2.4.8.3.1.组件窗口.....	21
2.2.4.8.3.2.构建脚本.....	21
2.2.4.8.3.3.事件图	21
2.2.4.8.3.4.功能	22
2.2.4.8.3.5.变量	22
3. 目标.....	2.3
3.1.最终学位项目的主要目标。	23
3.2.目标分解.....	2.3
4. 方法.....	24
4.1.开发方法.....	24
4.2.项目管理	25
4.3.版本控制和存储库.....	25
5. 工作内容	26
5.1.视频游戏设计文档 (GDD)	26
5.1.1.游戏的一般术语.....	26
5.1.1.1.论据摘要.....	27
5.1.1.2.功能集	27
5.1.1.3.性别.....	27
5.1.1.4.观众.....	28
5.1.1.5.游戏流程总结.....	28
5.1.1.6.游戏外观.....	29
围.....	30
5.1.2.游戏玩法和机制	31
5.1.2.1.游戏玩法	31
5.1.2.1.1.游戏目标	31
5.1.2.1.2.进展.....	31
5.1.2.1.3.任务和挑战结构.....	32
5.1.2.1.4.角色动作	38
5.1.2.1.5.游戏控制.....	39

5.1.2.2。力学	40
5.1.2.3。游戏选项.....	48
5.1.2.4。重播和保存.....	49
5.1.3.历史、特征和人物	54
5.1.3.1。历史.....	54
5.1.3.2。游戏世界	55
5.1.3.2.1。出发岛.....	56
5.1.3.2.2。图腾岛 1.....	56
5.1.3.2.3。图腾岛 2.....	57
5.1.3.2.4。图腾岛 3.....	58
5.1.3.2.5。图腾岛 4.....	58
5.1.3.2.6。图腾岛 5.....	59
5.1.3.2.7。图腾岛 6.....	59
5.1.3.2.8。图腾岛 7.....	60
5.1.3.2.9。图腾岛 8.....	60
5.1.3.2.10。镇岛	61
5.1.3.2.11。森林岛.....	62
5.1.3.2.12。岛决赛.....	62
5.1.3.3。主角。阿本	
宁.....	
5.1.3.3.2。 NPC 角色 2。图腾.....	65
5.1.3.3.3。 NPC角色3.长相老者的生物。	66
5.1.4.游戏关卡 Delfrydoland。	68
5.1.4.1。概括	68
5.1.4.2。介绍材料.....	68
5.1.4.3。目标.....	69
5.1.4.4。地图	69
5.1.4.5。道路.....	69
5.1.4.6。相遇.....	70
5.1.4.7。水平指南	70
5.1.4.8。结束材料.....	71
5.1.5.界面	71
5.1.5.1。平视显示器.....	71
5.1.5.2。菜单	73
5.1.5.2.1。主菜单.....	74
5.1.5.2.2。选项菜单。	75

5.1.5.2.3。暂停菜单.....	76
5.1.5.2.4。新游戏确认菜单.....	76
5.1.5.2.5。游戏结束菜单（当玩家死亡时）	77
5.1.5.3.相机	80
5.1.6.声音	80
5.1.7.帮助系统.....	81
5.1.8.人工智能	82
5.1.8.1.NPC AI（生命之 树、老年村民和最终岛图腾）	82
5.1.8.2.符文守护者图腾的 AI	83
5.1.8.3.魔像人工智能。	84
5.1.9.技术指南[25]	85
5.1.9.1.硬件.....	85
5.1.9.2.软件.....	85
5.1.9.3.游戏架构	85
5.2.开发和实施。	86
5.2.1.项目的创建.....	86
5.2.2.执行角色的动作。	92
5.2.2.1.相机旋转。	94
5.2.2.2.角色移动	97
5.2.2.3.上升/下降.....	101
5.2.2.4.跳过.....	101
5.2.2.5.干式制动	102
5.2.2.6.冲刺纠正.....	102
5.2.2.7.赛车中的涡轮增压.....	103
5.2.1.8.状态变更管理。	104
5.2.1.8.1.手动状态更改。	104
5.2.1.8.1.1.手动下降状态更改.....	104
5.2.1.8.1.2.手动更改航班状态.....	104
5.2.1.8.1.3.手动赛车状态更改.....	105
5.2.1.8.2.自动状态更改的管理.....	106
5.2.1.8.2.1.行走和跌倒之间的自动状态变化管理.....	107
5.2.1.8.2.2.状态更改补间.....	108
5.2.1.8.2.3.从飞行到下降的自动插值.....	109
5.2.1.8.2.4.从落到飞的自动插补.....	111

5.2.1.8.2.5.从赛车到飞行的自动插值.....	113
5.2.1.8.2.6.激活补间.....	115
5.2.1.9.使控制器与 Oculus Rift DK1 兼容.....	116
5.2.3.游戏级开发。	117
5.2.3.1.资产创建和导入管道	117
5.2.3.1.1.从 3ds max 导入资源	117
5.2.3.1.2.从另一个 UE4 项目导入资产	119
5.2.3.2.创建项目的土地（景观） 。	119
5.2.3.3.创建火火花粒子效果.....	121
5.2.3.3.1.粒子系统的剖析.....	122
5.2.3.3.2.实施火花系统。	123
5.2.3.4.添加照明。	131
5.2.4.游戏音效。	131
5.2.5.用户界面的实现	132
5.2.5.1.使用 Widget 蓝图创建用户界面的管道	133
5.2.5.2.游戏内使用	134
5.2.5.3.主菜单	134
5.2.5.4.选项菜单.....	135
5.2.5.4.1.修改屏幕分辨率.....	135
5.2.5.4.2.修改游戏总音量.....	136
5.2.5.5.暂停菜单	137
5.2.5.6.平视显示器.....	138
5.2.5.6.1.存货.....	138
5.2.5.6.2.振铃测试定时器	141
5.2.5.6.3.序幕、尾声和演职员表屏幕。	142
5.2.5.6.4.挑战通过和失败屏幕、游戏已保存以及您无法使用该项目.....	
5.2.6.游戏机制的实现。	143
5.2.6.1.传送器	144
5.2.6.3。 NPC 和图腾的对话	150
5.2.6.4.游戏结束时的大门。	154
5.2.7.游戏保存/加载系统。	157
5.2.7.1.保存游戏。	158
5.2.7.2.加载游戏.....	159

5.2.7.3。删除游戏.....	160
5.2.8。人工智能的实施.....	160
5.2.9。 Level Blueprint 163中游戏一般游戏流程的实现	
6.结论	167
6.1。改进建议和未来工作。	168
7. 参考书目和参考文献	170
7.1。介绍。	170
7.2.理论框架或现有技术.....	170
7.3.方法.....	171
7.4.视频游戏设计文档	172
7.5.开发和实施。	172

一、简介

随着岁月的流逝,人们开发了新的娱乐形式,因为
休闲的需要是人性本身的一部分。在这些形式中,我们发现
电子游戏自 1940 年代[1] 的历史开始以来,在追随者 (如游戏类型、游戏方式、开发团队数量和技术)方面都在
大幅增长,并持续数年超过

音乐和电影在西班牙并存,而且每年都在增加[2]。

目前,此外,鉴于开发视频游戏的大量技术,
平台的数量,以及所有人对技术的日益吸收
人们由于其成本的下降,有可能在视频游戏市场上找到
来自具有数百万美元成本的项目,例如 Destiny 的案例,其成本达到
3.8 亿欧元[3],拥有多个学科的专家开发团队,并致力于
努力了几年,实现了任何人都可以完成的零成本项目
家,并且可以积累数百万美元,例如 Flappy Bird for Android,它是由一个
一个人在他家住了几天,每天他为创造者董阮赚了几
90,000 美元[4]。

有了这一切,可以看出电子游戏的世界可以成为一个赌注
对任何多学科开发人员或小型开发团队都很感兴趣,并且
我们谈论多学科是因为视频游戏项目包括来自
编程、声音、设计,还有艺术 2D 和越来越多的 3D。这是
所以,这就是问题所在。作为多媒体工程师,我们必须能够
处理具有这些特征的项目的开发,特别是如果我们已经执行了
创造和数字娱乐的专业化,甚至处理现实的各个方面
虚拟[5]。

对于这个最终学位项目,我们将检查是否可以涵盖一个项目
多学科,例如 3D 视频游戏,为此我们将使用
视频游戏虚幻引擎 4。此外,我们想要这个项目,考虑到已经完成的行程
制造,提供与 Oculus Rift DK1 外围设备、虚拟现实眼镜的兼容性
大学目前可供学生使用。我们还将创建
游戏设计文档 (GDD) 的视频游戏,描述所有相同的设计要进行
后面。

2. 理论框架或现有技术

该块旨在根据其上下文分析视频游戏引擎，
比较目前存在的最流行的引擎，并且，
突出显示虚幻引擎 4 提供的实用程序和功能，这导致选择该工具来创建我们将要记录的项目。

需要强调的是，我们将使用视频游戏引擎来构建游戏，而建模和动画工具，音频编辑程序，

和其他用于编辑图形的，这些只为我们的资产（资源）的生产服务，它
通过我们将使用的引擎。正是出于这个原因，决定在理论框架中包括
作为一个主题，视频游戏引擎和视频游戏本身。

2.1. 视频游戏概念

我们将尝试首先建立视频游戏的定义，以便我们能够更好地理解我们正在处理的这项工作。

电子游戏虽然复杂，但基于一些简单的东西。 Bernard Suits 写道：“玩电子游戏是克服不必要的障碍的自愿努力”[6]。反过来，维基百科给了我们一个定义，称视频游戏“是一种电子游戏，其中一个或多个个人通过控制器与配备视频图像的设备进行交互”[7]。

考虑到游戏包含以下活动：

- 需要至少一名玩家。
- 它有规则。
- 有获胜条件。

对视频游戏的最准确定义可能是“通过视频屏幕玩的游戏”。

2.2. 制作视频游戏的引擎

2.2.1. 上下文中的引擎

在开始之前，有必要定义以下内容，什么是视频游戏引擎？

视频游戏引擎一词诞生于 1990 年代中期，指的是
早期的第一人称射击 (FPS) 游戏，尤其是著名的 Doom，由 id
软件。Doom 的设计具有非常明确的架构，具有出色的
核心引擎组件（如渲染系统）之间的分离

三维,碰撞检测,或音频系统),游戏的美术资源,
游戏世界,以及玩它们的规则。

当开发人员开始创建时,这种分离的价值变得显而易见
从该基本结构开始的游戏,但具有不同的艺术、世界、武器、车辆和
其他资产(资源),除了游戏规则之外,这一切都只做了微小的改变
到现有的“引擎”。这标志着“mod”社区的诞生,一群
个人游戏玩家和制作视频游戏的小型独立工作室
使用原始开发人员提供的免费工具包修改现有游戏。正是由于有了电子游戏引擎,设计师不再依赖设计师,
还可以添加或更改游戏的部分内容

很快,这在其发明之前可能会很昂贵[8]。

从id软件引擎开始,其他公司决定自建
视频游戏,正如Epic Games在1998年决定使用虚幻引擎[9]或Valve使用其
源引擎在2004年[10]和其他后来的公司。

2.2.2.市场上的视频游戏引擎比较

目前有非常广泛的引擎来制作视频游戏,包括2D和
3D,付费和免费,并为两者提供或多或少的平台兼容性
像制作视频游戏的人一样发展。

考虑到数量,将它们全部命名会很复杂,维基百科列出了一个很大的列表
按许可证分类的数量:

V · T · E	Game engines (list)		[hide]
Source port · First-person shooter engine (list) · Tile engine · Game engine recreation (list) · Game creation system			
Free software / open source	2D	Adventure Game Studio · Beats of Rage · Box2D · Chipmunk · Cocos2d · Digital Novel Markup Language · Flixel · Exult · Game-Maker · Gosu · Jogre · KiriKiri · Moai SDK · ORX · Pygame · Ren'Py · StepMania · Stratagus · Thousand Parsec · VASSAL · Xconq	
	2.5D	Aleph One · Build · Flexible Isometric Free · Id Tech 1 · Wolfenstein 3D	
	3D	Away3D · Axiom · Blender Game · Cafe · Crystal Space · Cube · Cube 2 · Delta3D · Dim3 · Genesis3D · GLScene · Horde3D · HPL 1 · Irrlicht · Id Tech 2 · id Tech 3 (Quake3) · id Tech 4 · JMonkey · Luxinia · OGRE · Ogre4j · Open Wonderland · Panda3D · Papervision3D · Platinum Arts Sandbox Free 3D Game Maker · PlayCanvas · PLIB · Python-Ogre · Quake · Nebula Device · RealmForge · Retribution · Torque 3D	
	Mix	Allegro · Construct Classic · Godot · Lightweight Java Game Library · Spring · Visualization Library	
Proprietary	2D	Clickteam Fusion · Coldstone · Construct 2 · Corona · CRX · Fighter Maker · Filmation · GameMaker · GameMaker: Studio · Garry Kitchen's GameMaker · Generic Tile · Gold Box · MADE · Mscape · M.U.G.E.N · NScripter · RPG Maker · Shoot the Bullet · Sim RPG Maker · Sound Novel Tsukuru · Southpaw · Stencyl · Vicious · Virtual Theatre · V-Play · Z-machine · Zillions of Games · ZZT	
	2.5D	Genie · INSANE · Infinity · Jedi · Pie in the Sky · Super Scaler · UbiArt Framework	
	3D	4A · Advance Guard Game · Anvil/Scimitar · Arsys · Beelzebub · Bork3D · BRender · C4 · Chrome · Creation · CryEngine · Crystal Tools · Dagon · Diesel · Digital Molecular Matter · Disrupt · Dunia · EAGL · EGO · Electron · Elfflight · Enforce · Enigma · Essence · Flare3D · Fox · Freescape · Frostbite · Geo-Mod · GoldSrc · HeroEngine · HydroEngine · HPL 2 · id Tech 5 · id Tech 6 · Ignite · Iron · IW · Jade · Kinetica · LS3D · Leadwerks · LithTech · Luminous Studio · LyN · Marmalade · Mizuchi · MT Framework · NanoFX GE · Odyssey · Orochi · Outerra · Panta Rhei · Phoenix Engine (Relic) · Phoenix Engine (Wolfire) · PhyreEngine · Q · Real Virtuality · REDEngine · Refractor · RenderWare · Revolution3D · Riot · RAGE · SAGE · Serious · Shark 3D · Silent Storm · Sith · Source · SunBurn XNA · Titan · TOSHI · Truevision3D · Unigine · Unity · Unreal · Vengeance · Visual3D · Voxel Space · XnGine · X-Ray · Yebis · YETI · Zero	
	Mix	CPAGE · Dark · Gamebryo · Hybrid Graphics · Kaneva Game Platform · Metismo	
Proprietary Game middleware (list)	AiLive · Euphoria · Gameware · GameWorks · Havok · iMUSE · Kynapse · Quazal · SpeedTree · Xaitment		

图 1. 引擎列表

来源:维基百科[11]

我们将重点介绍和比较 3 种最著名的发动机,用于生产

现在的电子游戏。这些是来自 Epic Games 的 Unreal Engine 4,来自 Unity 的 Unity

技术,y CryEngine de Crytek[15]。

2.2.2.1.虚幻引擎 4。



图 2. 虚幻引擎徽标

资料来源:维基媒体[12]

Unreal Engine 4,简称UE4,具有出色的图形能力,包括但不限于其他方面,先进的动态照明功能,以及可以在一个场景中一次处理多达一百万个粒子的粒子系统。

尽管虚幻引擎 4 是 UDK 的继承者,但 UDK 和 UE4 之间的版本变化确实很明显。为提高视频游戏制作的便利性而进行的更改。

在要突出显示的更改中,主要的更改之一是 UE4 的脚本语言,它在 UDK 中它是 UnrealScript 语言,现在它已完全被 UE4 中的 C++ 取代。此外,UE4 现在使用蓝图来编写图形脚本,这是 Kismet 的高级版本,具有无需使用 C++ 编程即可完全制作视频游戏。

虚幻引擎 4 有权从以下公司收取 5% 的视频游戏收入每学期的前 3,000 美元。

2.2.2.2.统一。



图 3. Unity 徽标

资料来源:维基媒体[13]

Unity 引擎提供了广泛的功能并具有简单的用户界面。去理解。此外,它具有多平台集成系统,可让您导出游戏对于几乎所有现有的,目前是 3D 开发的最佳选择 Android 平台,因为它的压缩工具允许视频游戏不会特别重,并且不会消耗过多的资源。

该游戏引擎与主要的 3D 建模和动画应用程序兼容,例如 3ds Max、Maya、Softimage、Cinema 4D 和 Blender 等,这意味着它支持读取这些程序导出的文件而没有问题。

与此相反,Unity 是一个支付引擎,尽管从 5.0 版解锁几乎之前只有专业版才有的所有功能,这方面虚幻4引擎出来了优势在于免费并为任何下载它的人提供 100% 引擎功能。

2.2.2.3 CryEngine。



CRYENGINE®

图 4. CryEngine 徽标

资料来源:维基媒体[14]

CryENGINE 是由 Crytek 公司设计的一个相当强大的引擎,它是与 FarCry 一起引入游戏世界的。

该引擎设计用于家用游戏机和 PC,包括索尼和微软的当前一代,即 PlayStation 4 和 Xbox One。

CryENGINE 的图形功能可以与虚幻引擎 4 相匹配,具有出色的照明、逼真的物理、先进的动画系统等。此外,同样与 UE4 一样,该引擎具有非常强大的编辑器和关卡设计功能。

另一方面,即使 CryEngine 是一个非常强大的引擎,它的学习曲线开始有效地使用游戏引擎有点棘手。如果我不知道打算制作具有 AAA 级视觉特征的游戏 (通常是数百万美元的作品),通常不建议使用它。

CryEngine 有一个订阅许可证,每月收费 9.90 欧元/美元。对于大型项目范围内,有必要联系他们以获得允许访问 100% 引擎代码和 Crytek 直接支持的许可证。

2.2.3.发动机的选择

如果我们在开发人员层面问自己选择哪个引擎的问题,我们将面临一个需要考虑的决定,因为它有一个重要的学习曲线,而且它也是有必要熟悉许多工具。这就是为什么建议选择最适合我们需求的那个,但是,考虑到这一点很方便一系列细节[16]:

- 提供良好的文档。
- 它拥有一个没有被抛弃的良好用户社区。
- 考虑我们是否要修改引擎。

对于这个项目,我们选择使用虚幻引擎 4,除了提供其功能 100% 免费,让我们轻松获得一些成果相当不错,并且尽管已经出来了,但仍然拥有当前的用户社区 2014 年,目前几乎可以解决任何可能出现的疑问,包括蓝图技术,它允许我们在理论上开发一个完整的视频游戏 100% 无需编程,我们将以此为基础进行整个开发。

2.2.4.虚幻引擎 4 的架构概念及其脚本编程系统蓝图

接下来我们将对引擎核心的特点做一个简单的介绍,并介绍一些架构的概念[17]。稍后,您将制作一个介绍虚幻引擎 4 的图形脚本系统 Blueprints[19]。

2.2.4.1。 UObjects 和 Actors

Actors 是派生自 AActor 类的类的实例;所有的基类可以在游戏世界中定位的对象。

对象是继承自 UObject 类的类的实例;所有人的基类虚幻引擎中的对象,包括演员。所以,实际上,所有实例虚幻引擎是对象;尽管如此,Actor一词通常用于指代到在其层次结构中派生自 AActor 的类的实例,而术语 Object 用于引用不从 AActor 类继承的类实例。大多数我们创建的类中的一部分将在其层次结构中的某个点从 AActor 继承。

一般来说,参与者可以被解释为项目本身或实体,而对象是更专业的部分。 Actor 经常使用组件,这些组件是专门的对象,用于定义其功能的某些方面或存储的值属性的集合。以汽车为例,汽车就是 Actor 本身,而汽车的零件,例如车轮和车门,可能是组件那个演员的。

2.2.4.2。游戏性和框架类

基本的游戏类包括代表玩家、盟友和敌人,以及通过玩家输入或智能控制这些化身人造的。还有一些用于为玩家创建 HUD (平视显示器)和相机的类。最后,GameMode、GameState 和 PlayerState 等游戏玩法类建立了游戏规则,并允许您跟踪游戏和玩家的情况

他们进步。

所有这些类都是 Actor 的类型,可以通过编辑器,或在必要时生成 (使它们出现)它们。

2.2.4.3。代表世界上的玩家、朋友和敌人

2.2.4.3.1。典当

Pawn 是可以成为世界中的“代理”的 Actor。棋子可以被附身由控制器创建,因为它们被创建为容易接受输入,并且可以执行每个人自己的行动。请记住,Pawn 不一定是类人动物。

2.2.4.3.2。特点

Character 是人形风格的 Pawn。它带有一个用于碰撞的 CapsuleComponent，并且默认 CharacterMovementComponent。您可以执行基本的人类操作，例如 move，具有网络功能，并具有一些动画功能。

2.2.4.4。通过玩家输入或人工智能控制 Pawn。

2.2.4.4.1 控制器

控制器是可以负责指挥 Pawn 的 Actor。通常来自两种形式，AIController 和 PlayerController。控制器可以“拥有”一个 Pawn 并拥有控制他。

2.2.4.4.2。播放器控制器

PlayerController 是 Pawn 和控制它的人类玩家之间的接口。这 PlayerController 基本上完成了玩家告诉它做的事情。

2.2.4.4.3 人工智能控制器

AIController 是一种可以控制 Pawn 的人工智能。

2.2.4.5。显示玩家信息

2.2.4.5.1。平视显示器

HUD 是许多游戏中常见的“平视显示器”或 2D 屏幕。在里面可以显示生命值、弹药等。每个 PlayerController 通常都有一个平视显示器。

2.2.4.5.2。相机

PlayerCameraManager 是眼睛并管理它的行为方式。每个 PlayerController 通常都有其中一个。

2.2.4.6。建立后续行动和游戏规则。

2.2.4.6.1。游戏模式

“游戏”的概念分为两类。 GameMode 类是游戏的定义，其中包括它的规则和胜利条件等方面。它只存在于服务器上。没有应该有很多在游戏过程中发生变化的数据。

2.2.4.6.2。游戏状态

GameState 类包含游戏的状态，其中可能包括诸如连接球员，得分。这是国际象棋中的棋子或任务列表。这必须在开放世界游戏中完成。 GameState 存在于服务器和所有客户端，并且可以自由复制信息以保留所有机器更新。

2.2.4.6.3。玩家状态

PlayerState 是游戏参与者的状态，例如人类玩家或机器人正在玩模拟玩家。作为游戏一部分存在的 AI 不应具有 PlayerState。什么适合在 PlayerState 中的示例数据包括玩家的姓名、他们的得分，或者如果您在夺旗游戏中携带旗帜。有一堂课每个机器上存在的每个玩家的 PlayerState，并且可以通过网络自由复制以保持一切同步。

2.2.4.7。框架类的关系

下图说明了所有这些核心游戏类如何相互关联。游戏由 GameMode 和 GameState 组成。加入的人类玩家游戏与 PlayerControllers 相关联。这些 PlayerController 允许玩家在游戏中拥有 Pawns，因此能够在关卡中具有物理表示。播放器控制器它们还为玩家提供输入控件、HUD 和 PlayerCameraManager 来管理相机视图。

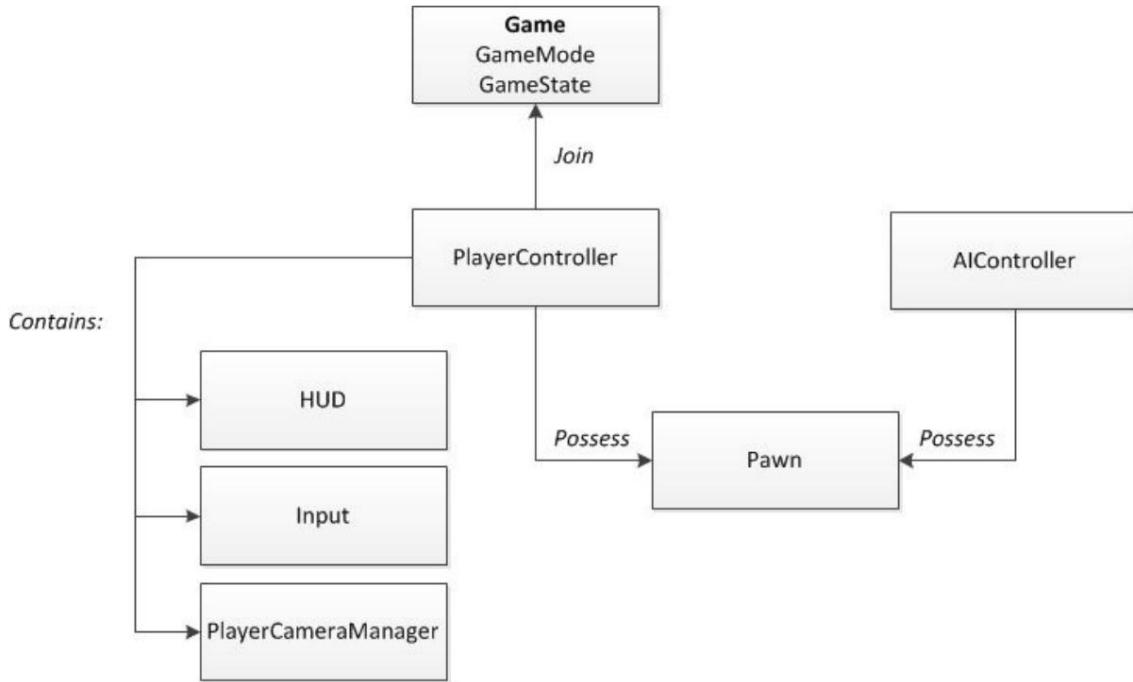


图 5. Unreal Engine 中的类关系图。

字体。虚幻引擎网站[18]

2.2.4.8. 蓝图可视化脚本

虚幻引擎中蓝图的可视化脚本是一个完整的游戏系统

使用基于节点的界面的概念实现以创建游戏元素

通过虚幻编辑器。这个系统非常灵活和强大,因为

为设计人员提供了使用几乎所有概念的能力

和以前只有程序员才能使用的工具。

通过使用蓝图,设计人员可以进行原型设计、实施或修改

几乎任何游戏元素,例如:

- 游戏。游戏规则、胜利条件等。
- 球员。修改和创建网格和材质或自定义角色。
- 相机。原型相机视角或更改相机

在游戏过程中动态。

- 输入（输入）。更改玩家控制或允许玩家通过
对项目的控制。

- 项目。武器、法术、触发器等。

- 环境。创建随机资产或程序生成项目。

2.2.4.8.1. 总体蓝图

蓝图是提供基于节点的直观界面的特殊资产，可用于创建新类型的演员和脚本级别的事件，为游戏设计师和程序员提供创建和迭代的工具。无需编写代码即可通过虚幻编辑器快速玩游戏。

2.2.4.8.2. 蓝图的类型

蓝图可以有多种类型，每种类型都有自己的功能，我们将定义我们打算在项目中使用的最重要的那些。

2.2.4.8.2.1. 课程蓝图

蓝图类，通常缩写为蓝图，是一种允许内容创建者可以快速向现有游戏类添加功能。这蓝图是在虚幻编辑器中以可视方式创建的，而不是通过键入代码，并保存为内容包中的资产。他们基本上定义然后可以作为实例定位在地图上的新的演员类别或类型行为与任何其他类型的 Actor 一样。

2.2.4.8.2.2. 关卡蓝图

LevelBlueprint 是一种特殊类型的蓝图，用作全局节点图。的水平。项目中的每个关卡都有自己的默认创建的关卡蓝图，可以通过虚幻编辑器进行编辑。无法通过界面编辑器。

与关卡本身或整个关卡中的特定 Actor 实例有关的事件，用于以调用函数或操作的形式触发动作序列的控制流。这与 Kismet 在 UDK 中使用的概念基本相同。

LevelBlueprint 还为流式传输和 Matinee 提供了控制机制用于与位于关卡内的参与者的事件进行通信。

2.2.4.8.2.3. 蓝图界面

蓝图接口是一个或多个功能的集合，可以添加到其他蓝图。任何具有接口的蓝图都应该具有这些功能。

接口的功能可以在每个拥有它们的蓝图中提供功能。添加。这就像编程中接口的概念，允许不同的类型

从公共接口共享对象和访问变量。为简单起见,蓝图

接口允许不同的蓝图相互共享和发送数据。

蓝图界面可以由内容创建者通过

与其他蓝图类似的方式,但它们有一些无法做到的限制

在他们中:

- 添加新变量。

- 编辑图表。

- 添加组件。

[2.2.4.8.3. 蓝图剖析](#)

蓝图的功能由几个元素定义,其中一些是

默认显示,而其他的可以根据我们的添加

需要。这使我们能够定义组件、改进初始化和

初始化操作,响应事件,组织和模块化操作,定义

属性等

[2.2.4.8.3.1. 组件窗口](#)

随着对组件是什么的理解,组件窗口中的

蓝图编辑器允许我们向蓝图添加新内容。这为我们提供了方法

添加碰撞几何体,例如 CapsuleComponents、BoxComponents 或 SphereComponents,

以 StaticMeshComponent 或 SkeletalMeshComponent 的形式添加几何体,使用 MovementComponents 控制运动

等。添加到组件列表中的组件也可以分配给实例变量,以提供对

它们在您的蓝图图表或其他蓝图中。

[2.2.4.8.3.2. 构造脚本](#)

构建脚本会在组件列表中启动

创建蓝图类的实例。包含一个被执行的图节点

允许 Blueprint 类的实例执行初始化操作。是

功能对于修改网格和材料属性或绘画很有用

世界上的笔画。例如,带有灯光的蓝图可以确定什么类型的地形

从一组网格中找到并选择要使用的正确网格。

[2.2.4.8.3.3. 事件图](#)

蓝图的 EventGraph 包含使用事件和函数调用的节点图

执行操作以响应与蓝图关联的游戏事件。这

EventGraph 用于添加所有蓝图实例共有的功能。

此外,它是建立交互性和动态响应的地方。例如,一盏灯

蓝图可以通过关闭来响应损坏事件,这种行为将是

我们根据这种类型的蓝图创建的所有灯光,因此当它们相应的伤害事件发生跳跃时,它们就会熄灭。

2.2.4.8.3.4. 功能

函数是拥有特定蓝图的图的节点,并且可以是

通过蓝图从另一个图表执行或调用。函数有一个点

由与包含单个执行引脚的函数同名的节点指定

出口。当从另一个图中调用一个函数时,输出执行引脚为

激活,这使我们可以继续执行脚本逻辑。

为了更清楚地解释它,如果我们有一个功能,我们将有一个输入和输出引脚。

输出,输入引脚是要激活的功能的信号,而输出引脚是

另一个组件执行的信号,这样我们就可以创建执行链

容易地。

2.2.4.8.3.5. 变量

变量是存储值或对世界中的 Objects 或 Actors 的引用的属性。我们将能够通过蓝图在内部访问这些属性

包含,或者可以发布它们以便能够访问它们并从其他人修改它们的值

蓝图。

3. 目标

3.1. 最终学位项目的主要目标。

该项目的目标是使用 PC 实现 3D 视频游戏
虚幻引擎 4 视频游戏，并将其整个开发集中在可视蓝图脚本系统上。

有了这个目标，它打算进行电子游戏的先前设计，分析特征
由引擎提供，并了解其操作和必要的管道
将资产包含在其中。

3.2. 目标分解。

以下是 TFG 以更具体任务的形式实现的目标：

1. 制作视频游戏的GDD（游戏设计文档）。
2. 涵盖从引擎中导入资产所需的管道方面
欧特克 3ds Max。
3. 创建菜单和 HUD 系统。
4. 使用虚幻编辑器提供的工具创建游戏世界。
5. 使用蓝图实现整个游戏逻辑。
6. 使游戏兼容 Oculus Rift DK1 虚拟现实设备。
7. 播放视频游戏。

4. 方法论

本节涵盖项目管理和开发方法的各个方面。

作为额外的,添加了一个小节来处理版本控制和存储库的各个方面。

4.1. 开发方法

为该项目的开发选择的开发方法是看板[20],这是一种基于带有标签的板的敏捷方法。

做出这一决定的原因是缺乏对发展的强加,

它提供了方法、定向到它所针对的人数（通常是小团队）,以及充分控制工作量的设施。此外,由于它是一种敏捷方法,它允许在特定阶段进行不太明显的开发。

并且比其他类型的更封闭的方法或某些更容易支持更改

设计规范过于严格。

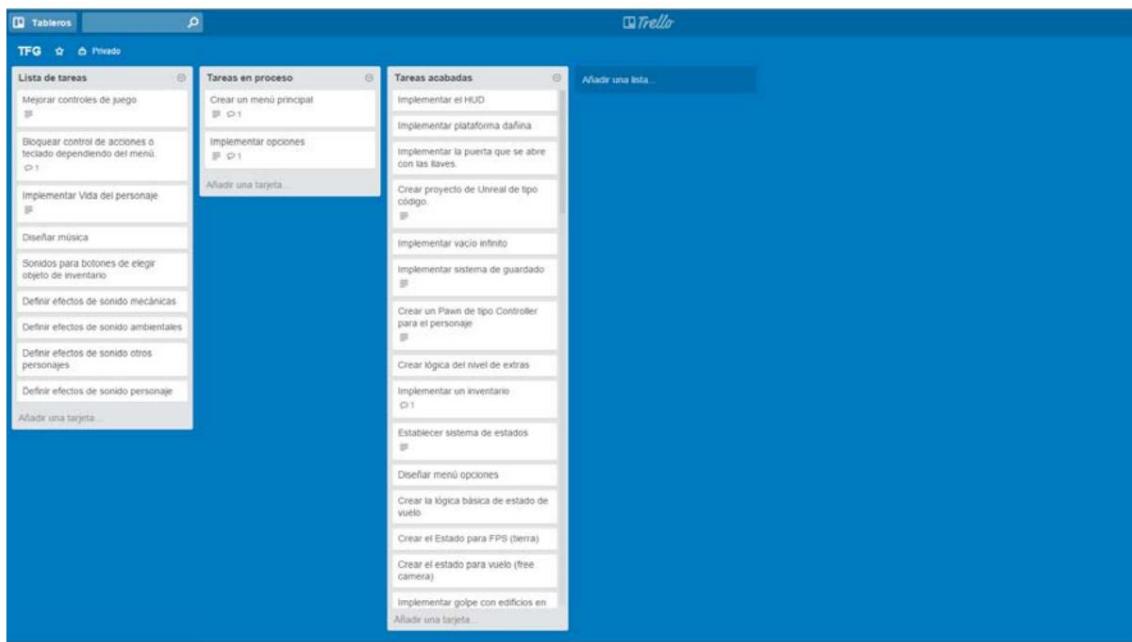


图 6. 项目看板组织的捕获

来源:自制。

此项目中的工作流程是向董事会添加新的待处理任务。

当打算执行一项或多项任务时,它们被放置在正在进行的任务列表中,当他们完成后,他们在已完成任务列表中排在最后。

4.2.项目管理

对于项目任务的管理,已决定使用 Trello 应用程序,它允许创建板并按标签组织任务。此应用程序中的任务支持定义任务或添加时方便的各种选项对这些的修改或注释（在我们的特定情况下以评论的形式,因为它只是一个人）。由于它是多平台的,它允许从智能手机或平板电脑,随时进行可能的更改或添加新任务。

4.3.版本控制和存储库

本作品基于虚幻引擎 4 中视频游戏的实现,同时具有
请记住,其创作的参与者人数仅为作者本人。
截至目前,版本控制尚未建立,因为一个虚幻项目
引擎 4 通常很重,定期上传这种性质的文件可以
需要相当长的时间,同时考虑到开发中没有更多成员,
只需要一个人来完成这个项目,最后,考虑到
相同的 UE4 引擎执行可编程的定期自动保存以避免数据丢失。
信息,已决定以某种方式限制项目的几个副本以确保安全
手动的。

5. 工作主体

开发工作主体分为两个主要部分。

1. 专门用于视频游戏设计文档 (GDD) 的第一个块,其中
它们涵盖了开发的视频游戏设计的所有方面。
2. 第二块,开发和实施,对应方面
研究项目制作的最相关技术人员。

5.1. 视频游戏设计文档 (GDD)



图 7. 视频游戏徽标。

来源:自制

该块详细描述了正确的所有必要的设计方面
开发的视频游戏的后续实施。这是太多的历史,经验
游戏玩法和关卡设计,例如用户界面和技术细节。

本文档基于主题中提供的 GDD 模板
电子游戏基础[21],多媒体工程专业第三年。

已决定给该项目起的名称是Arbennig:错误的世界。

5.1.1. 游戏一般



图 8. 视频游戏标题。

来源:自制。

5.1.1.1.论据摘要

我们移动到一个孤立而奇妙的世界,一个孤立的灵魂之地,体现在
图腾和其他文物,以及来自一些繁荣的灭绝文明和另一种新的生物。我们有
作为这个古老文明的一员从一棵树上出生,我们的物种不再是
属于这片土地。

然而,一切并没有失去,因为古代文明穿越到了另一个世界,这是可能的
与他们见面。为此,您必须进入被密封的异次元旅行室
并且只有获得8个符文才能进入,这些符文由分布在这片土地上的8个图腾保管。

图腾被灵魂附身,他们将给予我们他们守护的符文,以换取处理他们和他们的需求。

一旦获得所有符文,最终房间的封印将被打破,我们将可以带着旅行
我们的文明。

5.1.1.2.功能集

视频游戏的主要吸引人的特点如下:

- 有吸引力的视觉部分完美探索。
- 与Oculus Rift 外围设备兼容。
- 使用最近的Unreal Engine 4 引擎开发。

5.1.1.3.性别

Arbannig 的类型是开放世界的第一人称视频游戏 ,其中
拼图组件和技能。这种类型的这种类型的一个明显例子对应于
上古卷轴:天际之类的游戏。

5.1.1.4.观众

鉴于设计的特点,游戏在年龄等级之内
PEGI12型,根据PEGI官方网站提供的描述
PEGI12 [23]:



图 9. Pegi12 徽标。

资料来源:维基媒体[22]

“这一类别包括对幻想角色表现出更加形象化的暴力和/或对类人角色或可识别动物的非形象
暴力的视频游戏,以及表现出更形象化的裸体的视频游戏。粗话必须是温和的,不得包含性亵
渎。”

尽管如此,视频游戏仍适合任何年龄较大的人。

5.1.1.5.游戏流程总结

下面是基本游戏流程的示意图。在其中我们可以看到每个
游戏的组成部分之一,包括菜单、游戏和游戏本身。

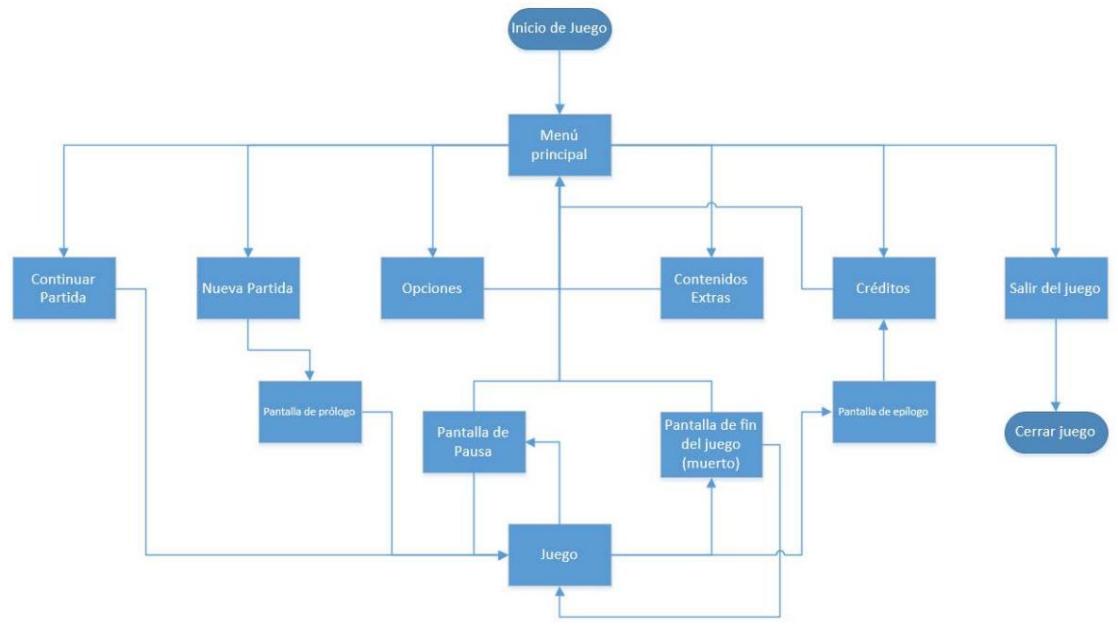


图 10. 游戏画面流程。

来源:自制。

这个游戏流程基本上包括:

1. 启动游戏应用。
2. 打开一个主菜单,有不同的选项,我们可以在其中继续游戏
(如果我们还没有,则禁用) ,或者开始一个新的,修改一些
配置参数,访问故事的额外内容,获得学分
开发,以及退出游戏的选项,这将使我们退出应用程序。将遵循
开始游戏以继续游戏流程。
3. 我们开始一个新游戏,我们可以 100% 击败游戏,这将
最终会导致通过最终房间的游戏积分屏幕。或者,另一方面
侧面,我们可以访问暂停菜单并返回主菜单。
4. 相反,如果玩家死亡,则会出现一个游戏结束菜单,提供选项
从上一个保存点再次播放,或返回主菜单。
5. 如果我们 100% 通关游戏,在获得学分后,这些将带我们,最后,
到主菜单。

5.1.1.6. 游戏外观

关于游戏体验,我们打算主要强调两个方面,
自由和幻想。

一方面,在发现周围环境时,给玩家一种自由的感觉。
飞越一个完全开放和完美探索的世界的能力是
要点。此外,作为第一人称的电子游戏这一事实导致了
更大的沉浸感和更大的自由感。

另一方面,幻想,游戏要求一个梦幻般的,田园诗般的世界,一个梦幻般的世界,一个
从游戏一开始你就会意识到你将成为游戏的一部分的空间
冒险。

游戏的配乐,包括音乐、环境声音和音效,都试图增强上述这些感觉。

5.1.1.7.范围

当我们发现自己在一个开放世界的电子游戏面前时,鉴于特定的特征,
这有,我们有一个单一的大的一般层次,一个它发展的公共空间
冒险和不同的挑战。这个单一的世界对应于德尔弗里多兰的土地。



图 11. 编辑器中的 Delfrydoland 地图。

来源:自制。

另一方面,关于我们将要遇到的角色,除了我们之外,只有NPC。
播放器。我们将找到的 NPC 对应于 3 种类型:

- 1.帮助NPC（父亲树）,他们将帮助我们控制和实现什么目标
玩家有。
- 2.冒险的NPC（图腾）,为此需要进行一些测试,我们
报酬。
- 3.通用NPC（该地方的新居民）,以更好地居住在这片土地上
Delfrydoland,他们不说话也不给出关于这个地方的任何通用信息。

5.1.2.游戏玩法和机制

本节描述了游戏玩法的两个方面,其中包括
玩家、游戏任务、游戏控制等,以及机制的各个方面
游戏,保存系统和更详细的游戏流程等。

5.1.2.1.游戏玩法

应该强调的是,在讨论游戏性问题之前,要知道它代表什么。一个
维基百科[24]提供了一个明确的定义:

“玩家在可玩的使用环境中以有效性、效率、灵活性、安全性,尤其是满意度达到特定游戏目标的程度”

我们可以总结出玩电子游戏的舒适程度。

5.1.2.1.1.游戏目标

玩家在游戏中的目标是克服呈现给他的所有挑战,要么
如上所述。

一旦所有这些都被克服,游戏就完成了。鉴于游戏的动态,其中
玩家可以按照他希望的顺序克服呈现给他的不同挑战,无论是在
飞行学校,就像那些角色一样,一旦最后一个被克服,游戏就不会启动结局
的挑战,但访问将通过通向学分的特殊房间解锁
当所有这些都完成时,游戏的结束,其学分象征着游戏的结束。
获得所有特殊游戏钥匙后,即可进入这个特殊房间。

5.1.2.1.2.进步

玩家在视频游戏中的进度以获得符文为标志
的图腾。

随着玩家完成任务或挑战,他将在电子游戏中获得更高的技能,并更多地了解他周围的世界和古代文明。

此外,在设法获得所有符文后,您可以使用它们进入允许您完成游戏的最终房间。

5.1.2.1.3.任务和挑战结构

至于游戏中的挑战,它们都有一个共同点,那就是由8个图腾构成。

散落在舞台周围。

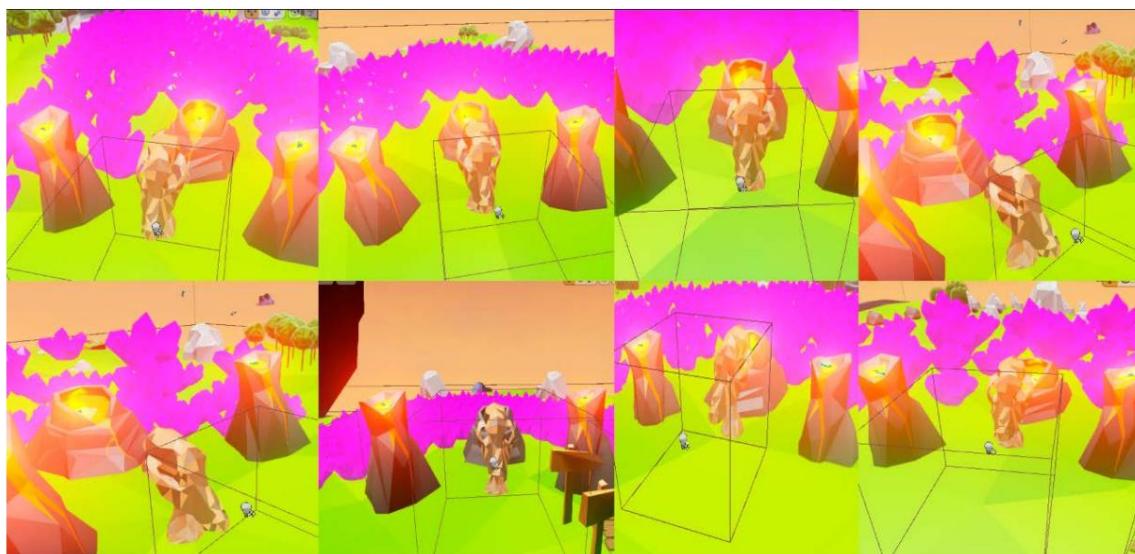


图 12. 来自编辑器的 8 个图腾的屏幕截图。

来源:自制。

此外,一次不能执行多个挑战,也就是说,当开始一个挑战时
图腾检查,在测试结束或对图腾说话之前无法进行进一步检查
意图取消考试。

游戏基于每个图腾具有以下挑战。

- 1.图腾1.听力挑战。在这个挑战中,只需要找到这个图腾,
因为他是第一个,和他说话,直到他说完他必须说的一切
说说游戏世界。

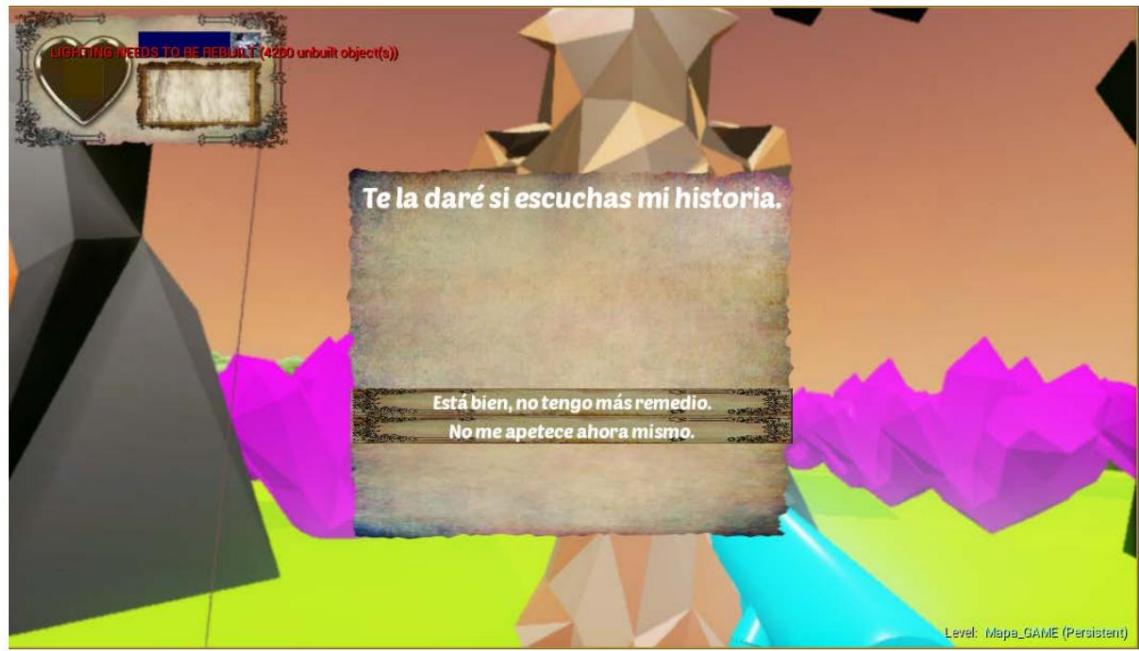


图 13. 捕获第一个图腾的挑战。

来源:自制。

2.图腾2.在小镇扩散。图腾命令我们向 9 个 NPC 传播信息

游戏中人口稠密的岛屿。为此,它为我们提供了 9 封邮件,我们必须交付它们,然后返回图腾通知他我们已经完成了挑战。



图 14. 第二个图腾挑战的捕获。

来源:自制

3.图腾 3. 清洁。在这个挑战中,图腾表明你的岛屿离一切最远世界,并且你想要的岛上出现了杂草 (10个灌木丛)让我们退出杂草必须放在一块木头上,表明,一旦全部移除,挑战就结束了。



图 15. 捕获第三个图腾挑战。

来源:自制。

4.图腾柱 4. 环挑战 1.舞台上设置了一系列环,通过这些环玩家必须在标记的时间用完之前通过 (通过其内部)。一度通过所有的箍,玩家克服了挑战。在没有完成挑战的情况下指定时间,玩家必须通过再次与图腾对话来重复挑战。



图 16。图腾 4 挑战捕获。

来源:自制。

5.图腾 5. 迷宫。在这个岛上,玩家必须通过迷宫进入地下,在它的尽头是图腾。这个挑战包括克服迷宫。



图 17。图腾 5 挑战捕获。

来源:自制。

6.图腾 6.追击。图腾命令我们跑到岛的另一端找到了,拿起一面旗帜,回去和他说话。在本次比赛中,选手会被几个追击魔像型 NPC 追赶,如果我们要迎接挑战。此外,不允许飞行,只能奔跑,如果飞行,玩家将失去挑战并被传送到入口。

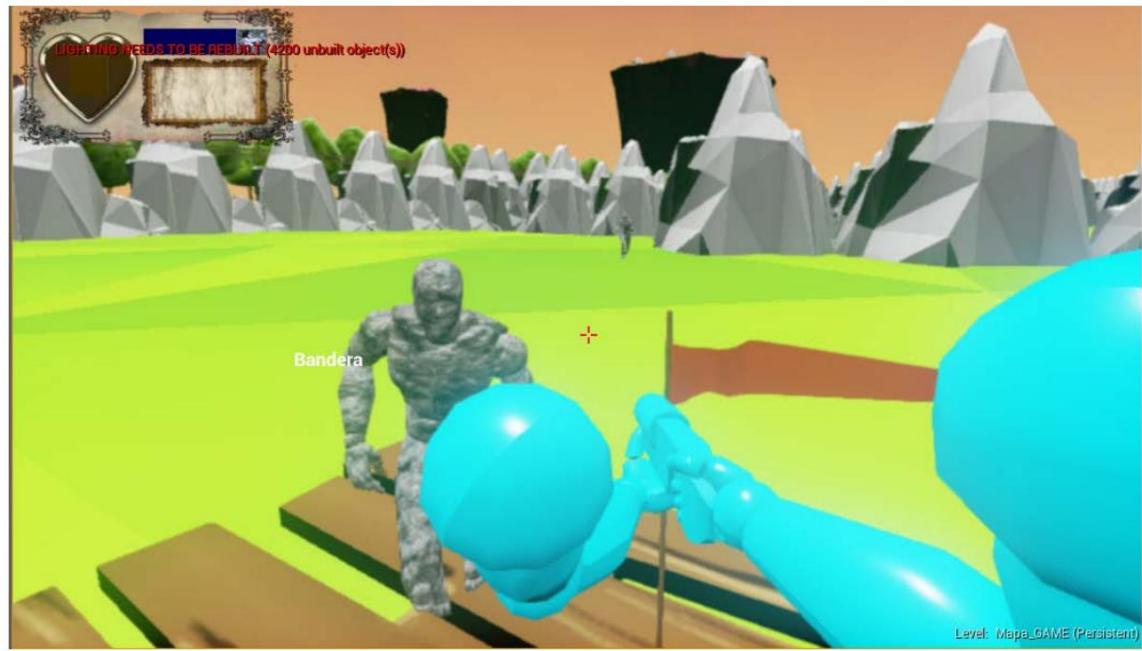


图 18.图腾 6 挑战捕获。

来源:自制。

7.图腾柱 7. 篮球挑战 2.跟游戏海岛挑战一样的操作。

图腾 4 但将耳环换成另一种类型。

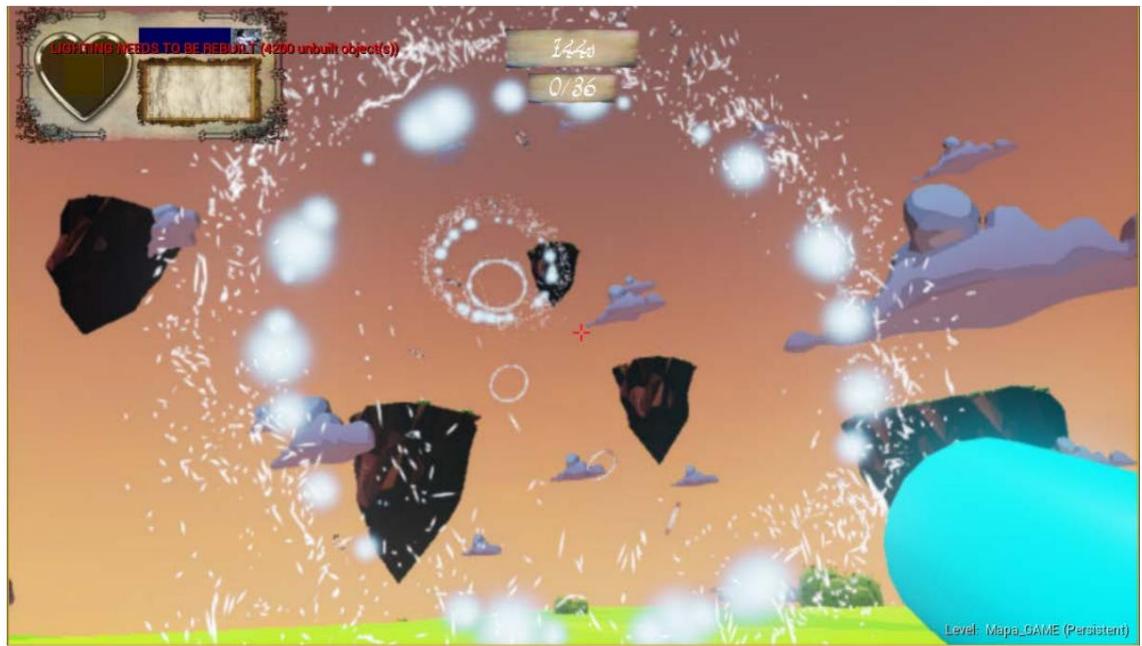


图 19. Totem 7 挑战捕获。

来源:自制。

8.图腾 8. 搜索。玩家必须进入世界南部的森林，

并找到放置在那里的旗帜。他必须弄清楚如何穿过森林（经过

通过没有植被的路径而不飞得很高）,因为如果它不通过路径合适,被传送到了森林的入口处。



图 20.图腾 8 挑战捕获。

来源:自制

关于游戏任务的最后一个方面,即进入最后的房间完成它,玩家可以探索世界并访问包含8 把锁的建筑。当您获得密钥时,它们将出现在您的HUD,当你得到它们时,你将能够与 8 个锁进行交互,这些锁现在将具有符文,阻挡通往房间的通道的门就会打开。



图 21. 游戏结束时放置符文打开的门的屏幕截图。

来源：自制。

5.1.2.1.4. 角色动作

角色可以在地面和空中执行动作。它具有三种状态

明显区分，您可以在它们之间自由切换：

基态。

在这种状态下，角色不能飞行，并且可以执行不同的动作：

一个。向您想要的所有方向移动（但不能向上或向下）。

i. 向前。

ii. 落后。

iii. 在右边。

iv. 向左。

b. 回转

i. Y 轴 ± 360°

ii. X 轴 ± 90°

c. 跳过。

d. 跑步。

e. 与环境互动。

i. 与其它角色交谈。

ii. 拾起/使用/与舞台上的非角色物品互动。

f. 激活航班状态

飞行状态。

在这种状态下，很像地面状态，玩家可以执行以下操作：

g. 朝你想要的所有方向移动。

i. 向前。

ii. 落后。

iii. 在右边。

iv. 向左。

v. 向上。

vi. 锯下。

g. 回转

哟。在 Y 轴上为 ±360°。

ii. X 轴±90°。

哟。与环境互动。

哟。与其他角色交谈。

ii. 拾起/使用/与舞台上的非角色物品互动。

J。激活基态。

◦激活快速飞行或赛车状态。

l. 干式制动。

米。暴跌。

2. 快速或赛车飞行状态。

在这种状态下,玩家的速度会逐渐增加,直到达到最大值,一旦达到最大值,它就会保持不变。可以的动作

执行播放器如下:

一个。移动 (不能向后移动,因为赛车状态让你移动

以越来越快的速度转发给玩家,此外,因为

自动向前移动,角色无法控制

向前移动) :

哟。向上。

ii. 下。

iii. 在右边。

四。向左。

湾。回转

哟。 X 轴±90°

ii. Y 轴±90°

C。进入飞行状态。

d。激活额外的涡轮。这个涡轮会消耗玩家的能量条,即

当涡轮未激活并通过箍时再生。

5.1.2.1.5. 游戏控制

可以使用键盘和鼠标进行演奏,并且可以使用或不使用 Oculus Rift。

系统会自动检测您启动游戏时是否已连接设备

oculus rift,如果是这样,无论我们是否使用鼠标,相机都可以由它控制。

另一方面,键盘和鼠标的游戏中控件,包括可选控件

Oculus Rift 子外设:

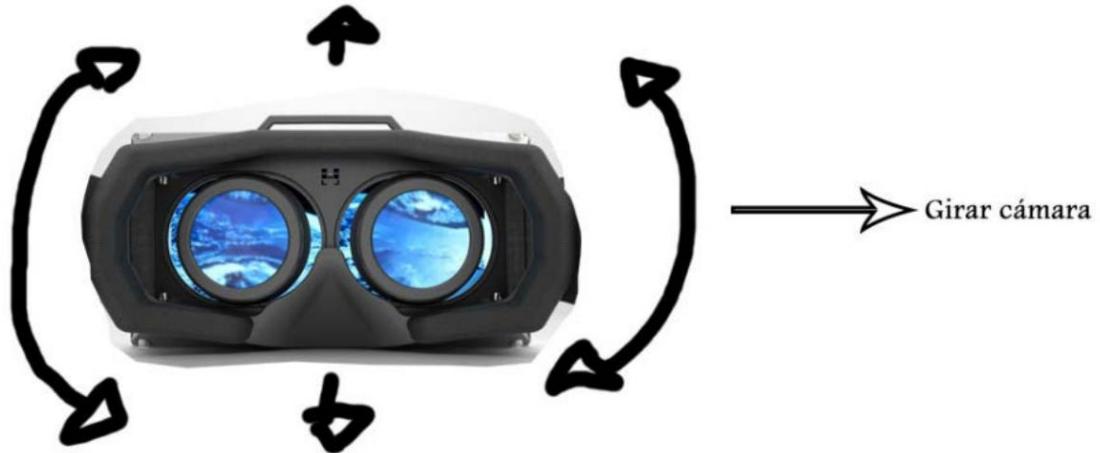


图 22. 在游戏中使用 Oculus。

来源:自制。

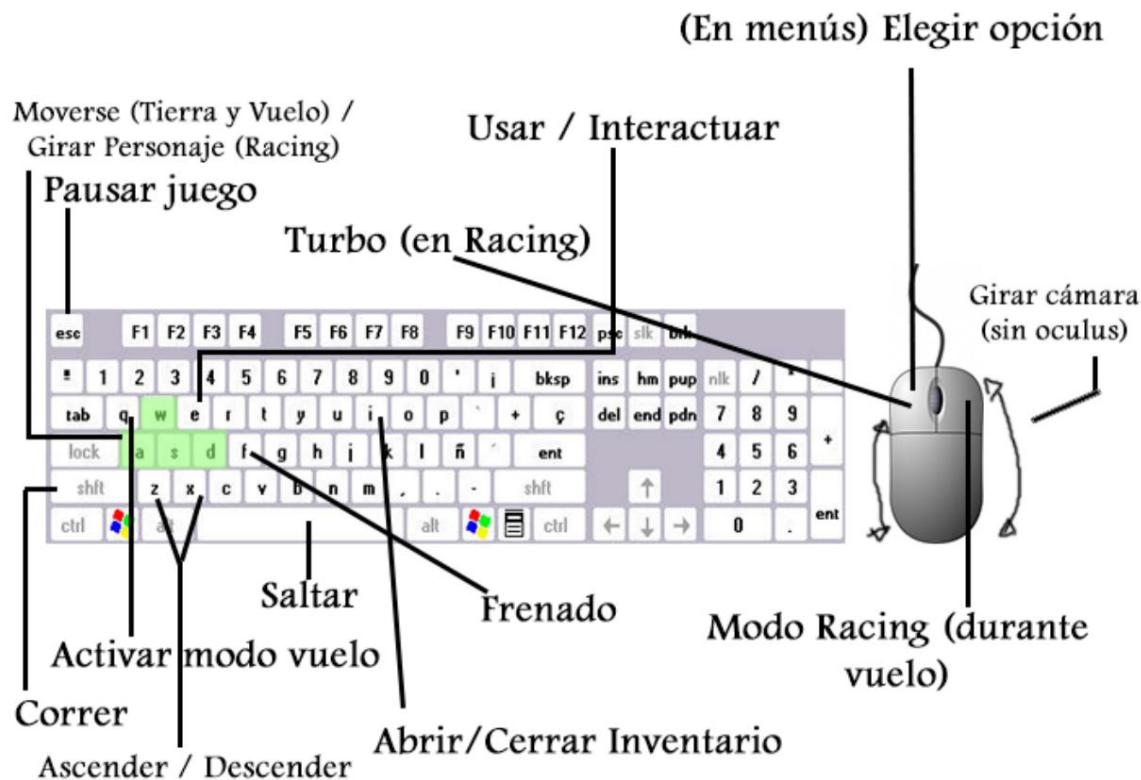


图 23. 带有键盘和鼠标的游戏玩家控件。

来源:自制。

5.1.2.2。机械的

我们所说的游戏机制是指玩家与之交互以创造或有助于游戏玩法,例如移动的平台,绳索摇摆,或滑冰。在本节中,除了游戏机制外,我们还将包括危害 (可以伤害或杀死玩家但没有智能的机制,例如破坏性平台),道具 (玩家捡起的物品以帮助他们游戏玩法,例如涡轮增压器) 和收藏品 (玩家捡起的物品,但它们不会对游戏玩法产生直接影响,例如游戏键)。

也就是说,在 Arbannig, 我们发现的机制如下:

机械的

传送器。



图 24. 分别用调试线绘制的传送器和传送目的地,从左到右。

如果玩家接触到该组件,他们会被传送到地图的另一个区域。

打开的门。

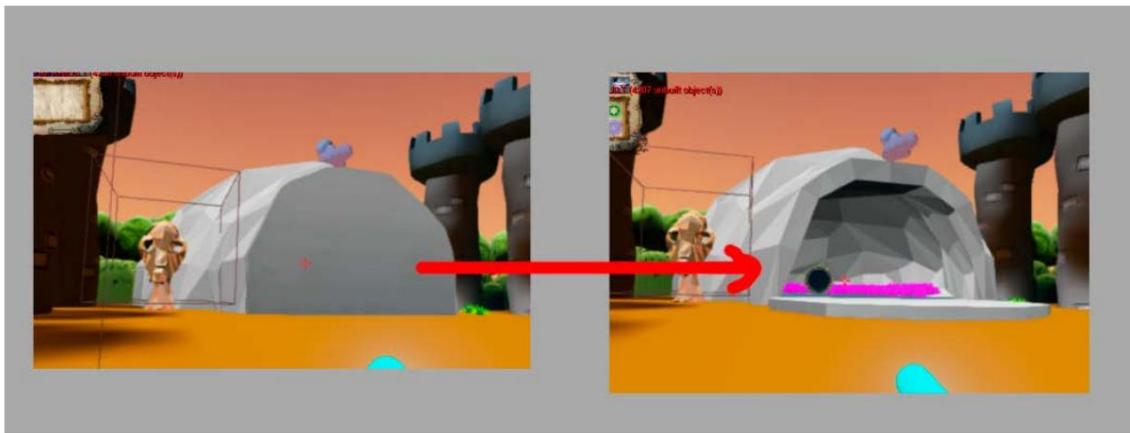


图 25. 关闭和打开的门捕获。

来源：自制。

通过满足某些条件或仅与它们交互来打开的门。

游戏结束时的大门。

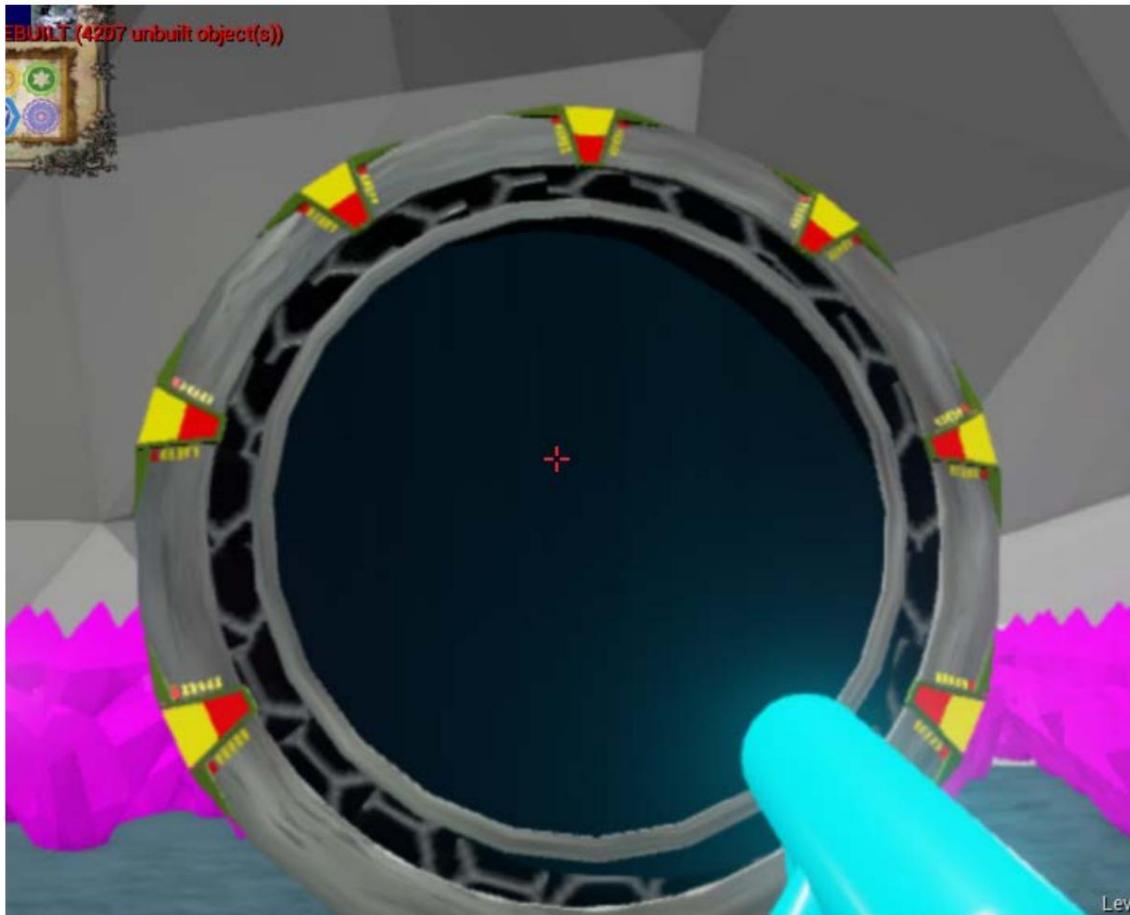


图 26. 最终栅极捕获。

来源：自制。

一旦我们与它互动或通过它,这扇门允许我们完成游戏。

危险

平台/有害物质。

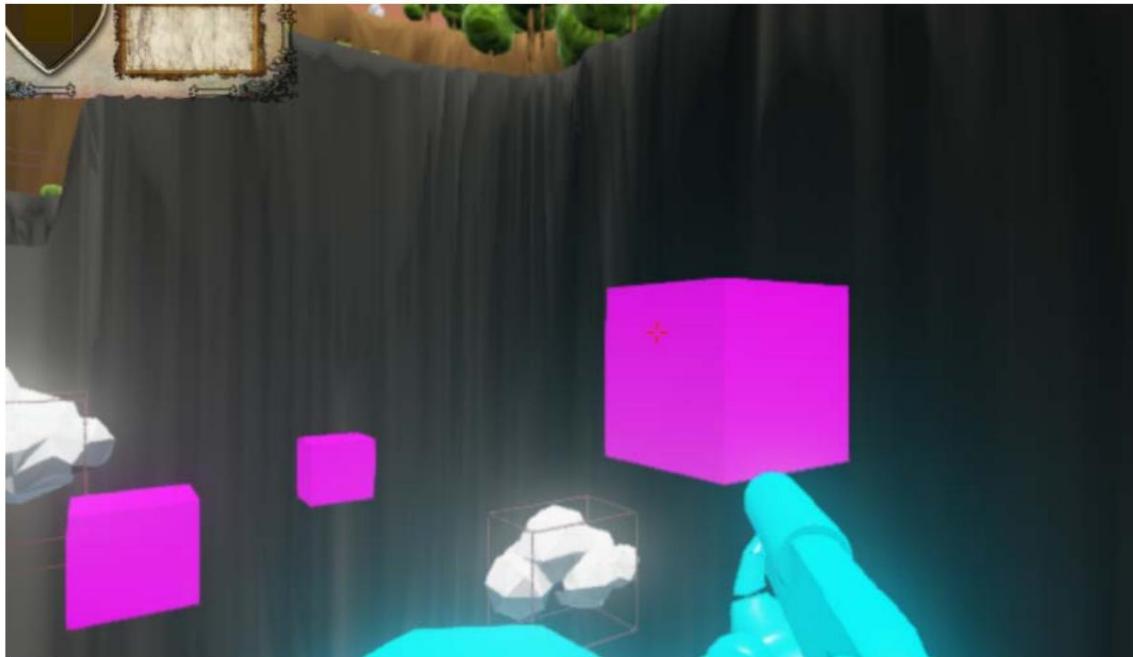


图 27. 捕获各种有害平台（紫色）

来源:自制。

如果玩家触摸它们,它们就会失去生命的组件。有可能在篮球的挑战中找到他们。
即死平台。



图 28. Game Sea Capture (Instant Kill Pad)

来源:自制。

玩家在与她接触时死亡。

通电

阿罗。

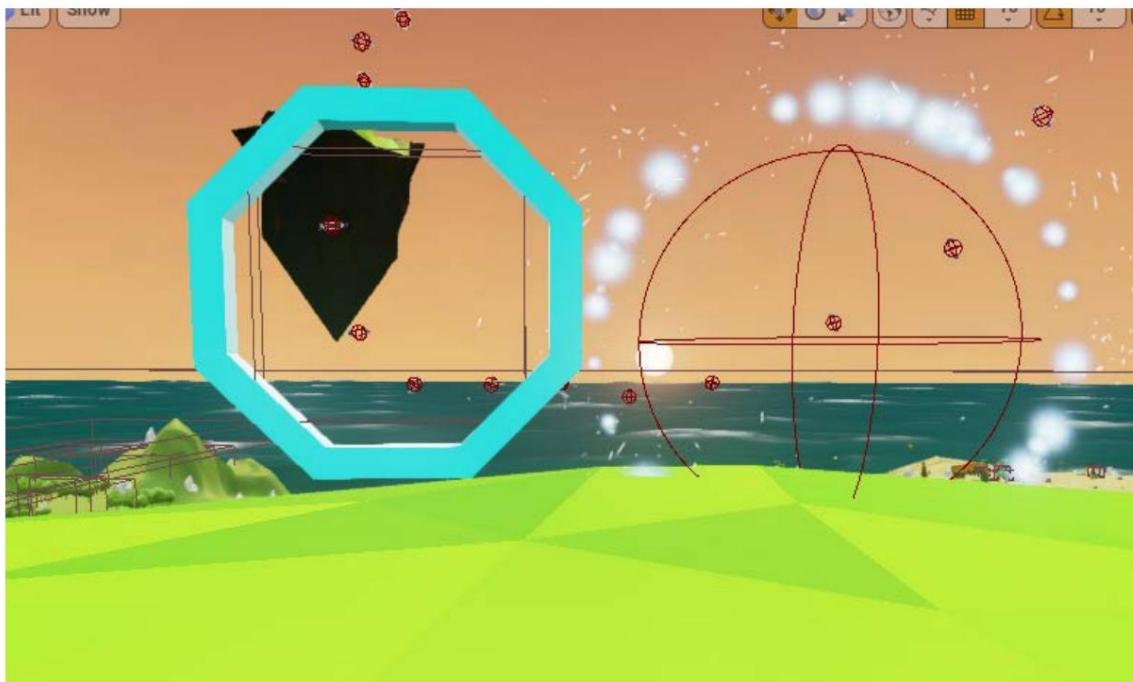


图 29. 游戏中两种类型的箍的捕获。

来源:自制。

为玩家提供涡轮增压的即时能量补充。

保存点。



图 30. 捕获保存点。

来源:自制。

为玩家提供涡轮增压器的健康和能量充电。

收藏品

演讲。

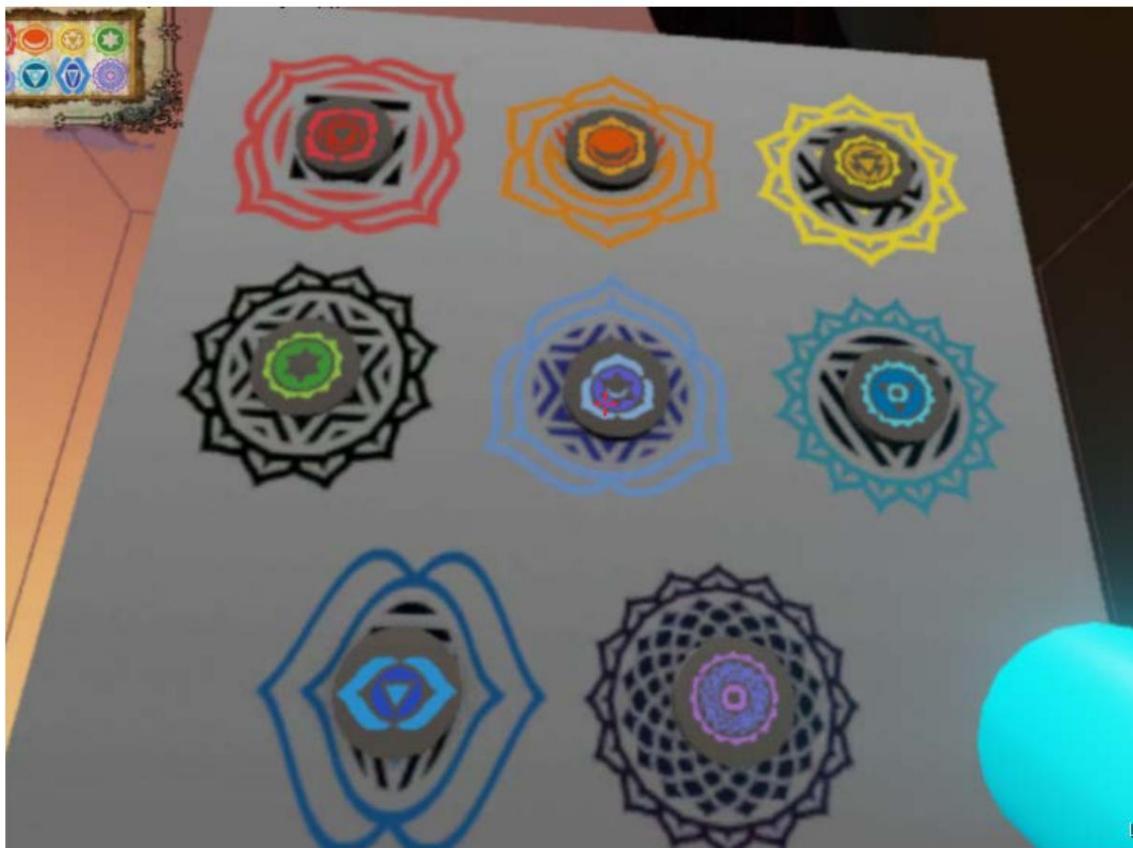


图 31. 捕获放置在最后一面墙上的 8 个符文。

来源:自制。

当玩家克服强加给他的测试和挑战时,他会得到一些特殊的钥匙,让他可以进入最后一个房间来完成游戏。您还可以找到

在世界上更多的钥匙。

岛屿图腾元素 2. 宣传册。



图 32. 信息手册的屏幕截图。

来源:自制。

玩家可以通过他们的库存将它们捡起来以便以后交付。

岛屿图腾元素 3.灌木。



图 33.擦洗捕获。

来源:自制。

玩家可以通过他们的库存将它们捡起来放到其他地方。

5.1.2.3。游戏选项



图 34. 捕获游戏选项。

来源:自制。

只能通过主菜单访问的游戏选项允许您修改视频和音频参数。

这些参数是:

- 修改游戏分辨率,让您在 4:3 格式和
16:9 格式。
- 修改游戏的总音量。

5.1.2.4。重播和保存

要保存游戏,可以访问分布在整个游戏中的不同保存点。

风景。



图 35. 捕获保存点。

来源:自制。

这些保存点存储玩家的进度（图腾通过）,并离开玩家
玩家在存档点所在的位置。

从桌面启动游戏时,如果
有,在主菜单中显示是否锁定继续游戏的选项（取决于是否有已保存的游戏）。



图 36. 主菜单的屏幕截图,其中继续被阻止的选项（没有游戏保存）。

来源:自制。

除此之外,在加载游戏附加内容的关卡时,它包含或多或少的附加内容根据玩家的进度。

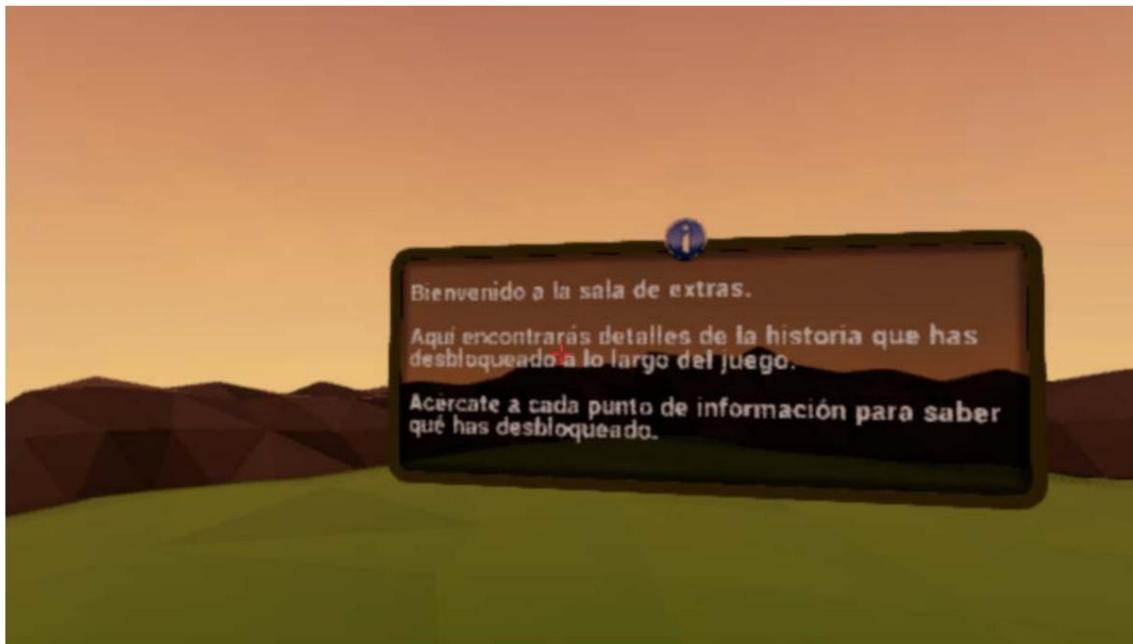


图 37. 捕获附加值。

来源:自制。

如果您在已保存游戏的情况下选择开始新游戏的选项,则
一旦玩家保存到新游戏,保存的游戏就会丢失,覆盖
上一个。玩家会收到此步骤的警告,以防他们无意中点击。



图 38. 新游戏确认菜单的屏幕截图。

来源:自制。

另一方面,如果玩家死亡,他们会被送到游戏结束屏幕,
这使您可以选择从上次保存的点再次播放。失去一切
保存游戏前取得的进展。

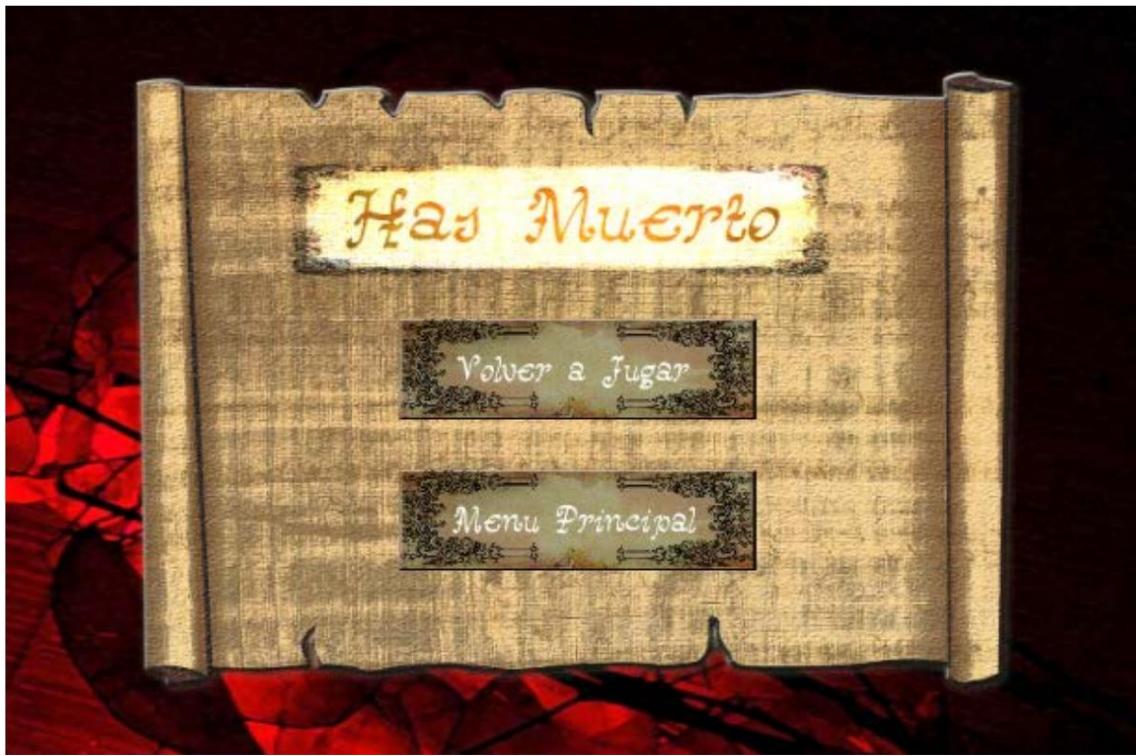


图 39. 游戏结束菜单的捕获。

来源:自制。

5.1.3.故事、特征和人物

5.1.3.1.历史

Delfrydoland的土地是一个神秘的地方。与世界隔绝时空
在外面,生活着一个对其过去、现在或未来知之甚少的社会。一个社会
由维持自然平衡的生命之树创造的众生,通过
他对生物的力量。

这个社会,长相如老人,长相一模一样,而且不吃东西,
他们不睡觉,不进行任何活动,不衰老,甚至没有名字或占据自己的位置
闲暇。他们通常住在一个岛上的村庄里
等待某些事件发生,这可能不是他们造成的。更重要的是,
他们甚至没有建造村庄,尽管这对他们来说并不重要。

在这种没有人问很多问题的环境中,生命之树
在德尔弗莱多兰创造了一个古代长者文明的成员,创造了一个
特别的。

阿本尼格属于决定离开世界的巴罗德文明

Delfrydoland 在其他世界寻求新的冒险。 Barod是创造者

生命之树 ,并在他们的物种发生几个世纪后确定了他们物种成员的创造时间

行军,以便他可以告知他们他们留下的世界,但为了

告诉他们,阿本尼格应该证明他有能力前往其他世界进行冒险

跟他们。也就是说,为了改变世界,他们在

在一个被8个魔法符文封印的石室里,这将被8个图腾保管,这些图腾将把它交给阿本尼格,以换取挑战。

多亏了图腾和一些仙女,阿本尼格设法发现巴罗德,一个非常繁荣、聪明的物种,渴望发现新世界,最终厌倦了

在他看来,当什么都没有的时候,Delfrydoland 的美妙但有限的世界,

他们可以在其中发现或实现,他们创造了一种特殊的技术,使他们能够前往另一个地方

更大的世界去探索。

最后,当 Arbannig 设法收集所有密钥并获得对这台机器的访问权限时

运输者,决定回到他的同类并进行许多新的冒险。

5.1.3.2. 游戏世界

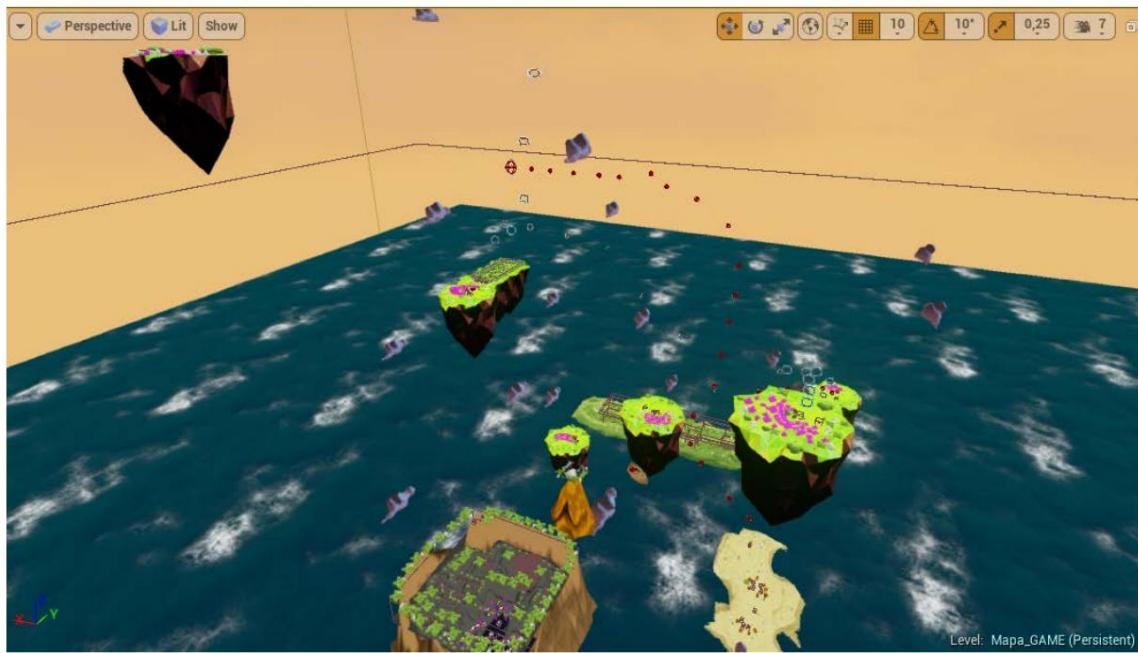


图 40. 从编辑器中捕获的游戏世界。

来源:自制。

游戏世界是位于海洋中央的一个多变的地方 ,其中有两个岛屿，就像天空中漂浮的小岛。我们总共发现了 12 个地层、5 个海岛和 7 个岛屿漂浮的。

5.1.3.2.1.启动岛。



图 41. 来自编辑器的家乡岛的屏幕截图。

来源:自制。

在这个岛上是玩家出生的地方 ,在这里我们找到了生命之树和一系列教我们如何玩耍的物品。

5.1.3.2.2.图腾岛 1.

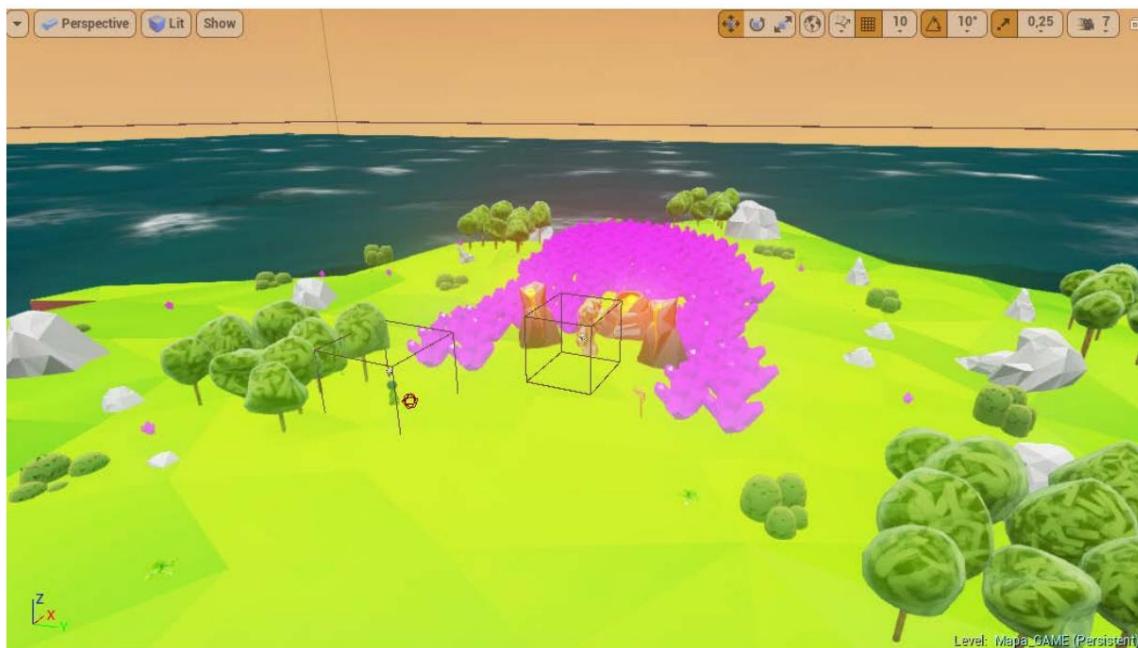


图 42. 来自编辑器的图腾岛 1 的屏幕截图。

来源:自制。

图腾1岛离起始岛最近,是一个浮岛。

5.1.3.2.3.图腾岛2。



图 43. 来自编辑器的图腾岛 2 的屏幕截图。

来源:自制。

就像图腾 2 岛一样,我们发现自己在另一个浮岛之前,在那里我们找到了

想要我们让你知道当它给我们符文时它会消失的蔑视图腾

门徒。

5.1.3.2.4.图腾岛 3.



图 44. 来自编辑器的图腾岛 3 的屏幕截图。

来源:自制。

图腾岛3是整个游戏中最远的岛屿,位于一端且远高度。

5.1.3.2.5.图腾岛 4.



图 45. 来自编辑器的图腾岛 4 的屏幕截图。

来源:自制。

在这个浮岛上,我们找到了派我们进行第一次飞行挑战的图腾。

5.1.3.2.6.图腾岛 5.



图 46. 来自编辑器的图腾岛 5 的屏幕截图。

来源:自制。

图腾岛 5 是由 Barod 创造的迷宫,充满了
带我们到出口的传送器,而图腾柱在传送器的尽头。

5.1.3.2.7.图腾岛 6.



图 47. 来自编辑器的图腾岛 6 的屏幕截图。

来源:自制。

图腾岛 6 是最大的浮岛,因为共有 3 个小岛,其中找到一个速度和能力的测试来躲避一些石头字符挑战。

5.1.3.2.8。图腾岛 7。



图 48. 来自编辑器的图腾岛 7 的屏幕截图。

来源:自制。

岛 7 包含另一个篮球挑战,比岛 4 上的挑战更复杂。

5.1.3.2.9。图腾岛 8。



图 49. 来自编辑器的图腾岛 8 的屏幕截图。

来源:自制。

8号岛是图腾所在的地方,它让我们在森林岛上寻找一面旗帜。

5.1.3.2.10。镇岛。



图 50.从编辑器中捕获的村庄岛。

来源:自制。

在这个岛上,我们发现了由树创造的新文明。房屋是巴罗德的房屋,
他们没有门,因为他们使用传送器进入。

5.1.3.2.11.森林岛。



图 51. 从编辑器中捕获的森林岛屿。

来源:自制。

森林岛是一个充满传送器的地方,要穿越它必须跟随
在它的地板上标有一条路径,如果我们想克服挑战,我们必须遵循
岛图腾柱 8.

5.1.3.2.12.岛决赛。



图 52. 从编辑器中捕获的最终岛屿。

来源：自制。

这个最后的岛屿是我们找到石室的地方，一旦我们拥有它就必须打开所有符文。

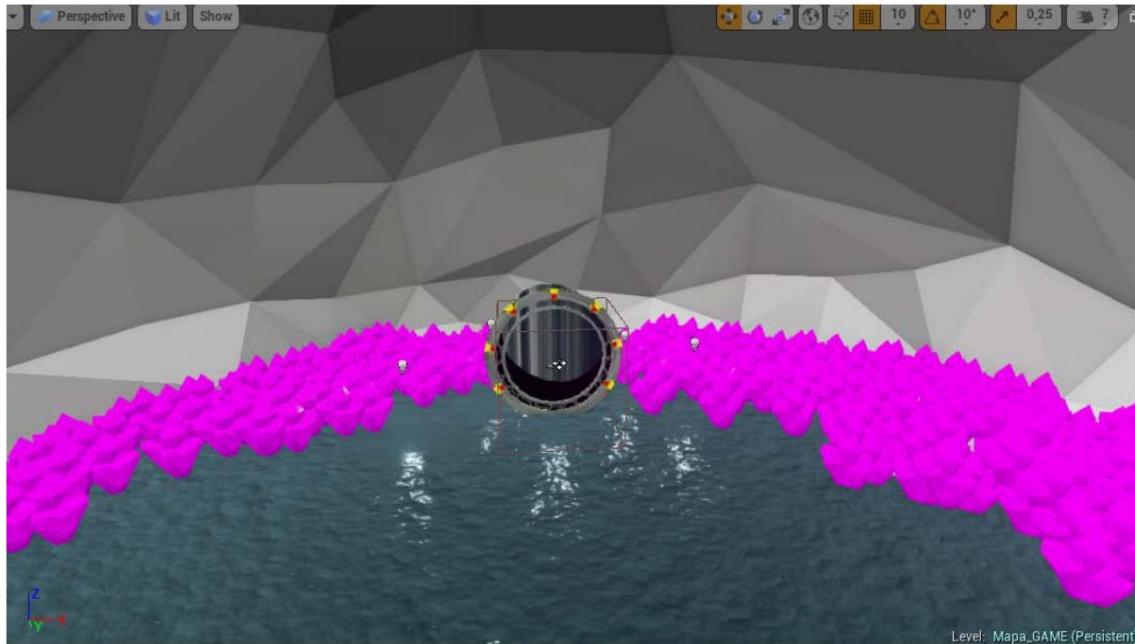


图 53. 从编辑器中捕获最终房间的内部。

来源：自制。

房间里面是传送器。

5.1.3.3。 Personaje 校长。特别的。

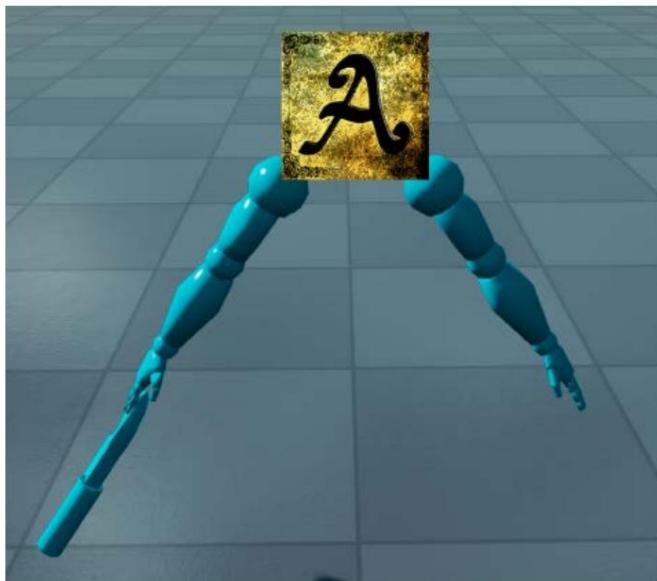


Figura 54. Special Personaje en primea persona.

来源:自制。

阿本尼格是生命之树创造的巴罗德。你必须对付世界的8个图腾才能能够获得让你与你的文明一起旅行的符文。

巴罗德在他离开后的几个世纪里为他的出生计时,从这样它就可以了解德尔弗里多兰的世界。

Arbannig 是可以执行角色动作 (5.2.1.4) 的角色,也是它具有涡轮飞行的生命和能量。

5.1.3.3.1。 NPC角色1.生命之树。



图 55. 生命之树的捕获。

来源:自制。

生命之树是巴罗德最伟大的发明之一。它可以创造生物和与德尔弗里多兰世界的任何存在进行心灵感应交流。他是创造者阿本尼格,也是一个老人模样的新文明。

5.1.3.3.2。 NPC 角色 2. 图腾。



图 56. 在调试模式下从游戏中捕获图腾。

来源:自制。

它们是由 Barod 创造的,用来储存符文,让 Arbannig 能够进入最后的传送室到另一个世界。它们都是一样的,尽管每个角色都有自己的个性。

当他们将符文交给阿本尼格时,他们的灵魂不再与图腾绑定,所以他们停止了能够与世界交流。

尽管如此,它仍然存在,一个从不离开身体的图腾,也没有可以放弃的符文,它是在最后一个房间找到的图腾柱,放置在那里向任何接近的人解释他怎么可能进入密封的房间。

5.1.3.3.3。 NPC角色3.长相老者的生物。



图 57. 以老者相貌捕捉众生。

资料来源:自己的阐述 (来自虚幻 mixamo 包的免费网格)。

这是德尔弗莱多兰居住的新文明,他们除了对想和他们说话的人说几句话外,没什么可说的,也没有太多事情要做。另外,它们看起来都一样。

5.1.4. 游戏关卡 Delfrydoland。



图 58. 游戏关卡高度。

来源:自制。

5.1.4.1. 概括

这个世界是电子游戏中唯一的一个。这是一个开放的世界,玩家可以探索并执行 8 个任务,让他可以进入游戏结束的最后一个房间。

在这里,我们可以找到上面描述的所有角色和地点,以及所有力学和任务。

5.1.4.2. 介绍材料

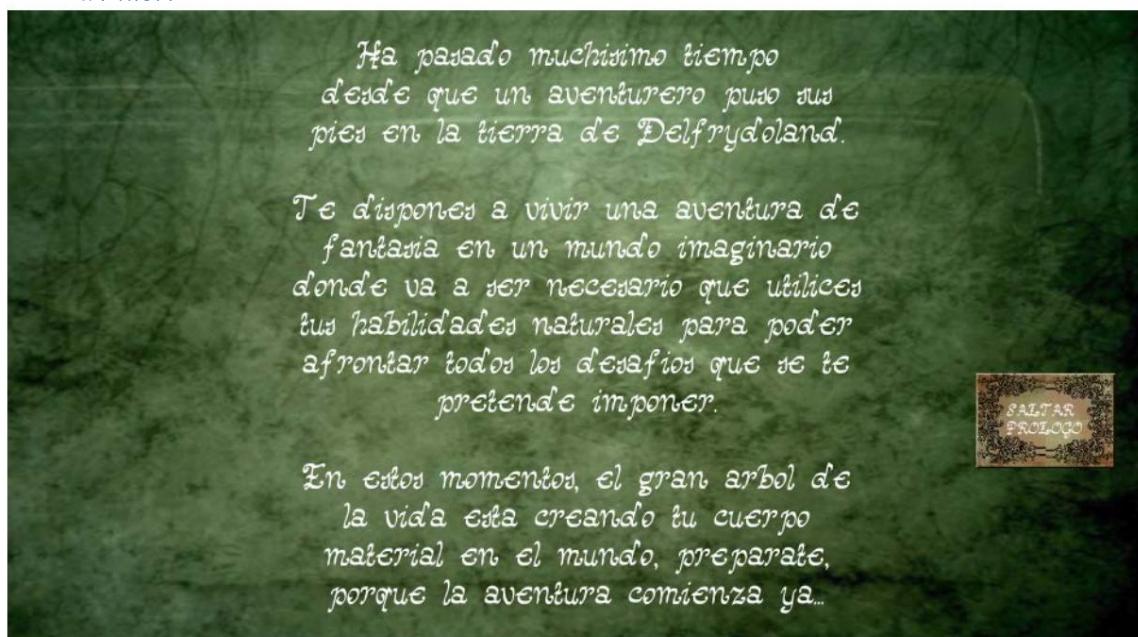


图 59. 序幕捕获。

来源:自制。

在访问该级别之前,会出现一个序幕屏幕,让我们知道它的历史很少。这个序言可以用一个按钮跳过。

除此之外,一旦游戏开始,玩家会发现自己在最初的岛上充满了帮助消息,让您了解游戏控制,以及有关世界。

5.1.4.3。目标

在 Delfrydoland 世界中要实现的目标是所有这些在关于任务和挑战结构 (5.1.2.1.3) 。

5.1.4.4。地图



图 60. 游戏地图。

来源:自制

5.1.4.5。道路

玩家,因为他可以执行飞行动作,并且因为他在一个世界中打开后,您可以按照您喜欢的任何顺序执行任何游戏任务。也就是说,从

它出现的起始岛屿,你可以自由地沿着你想要的路径前进,直到任何有图腾的岛屿来执行它所指示的挑战。

5.1.4.6。相遇

探索世界,有可能遇到章节中描述的不同角色字符数 (5.3.3)。也就是说,你将能够满足8个挑战图腾或岛屿图腾最后是生命之树,以及几乎在每在村子里,除了一个,在迷宫岛上。提到的每个角色有话要对玩家说,提供信息。

5.1.4.7。水平指南

因此,没有水平指南,如路径部分 (5.4.5) 中所述,玩家可以按所需顺序执行任何挑战,我们发现其中的逻辑在任务和挑战结构部分 (5.2.1.3)中描述。

一旦玩家完成每个挑战,他们就可以进入最后的房间,与放置符文的柱子互动,这将打开通往最终房间的通道门游戏,带有一个传送器,当通过它或与之交互时,使游戏学期。

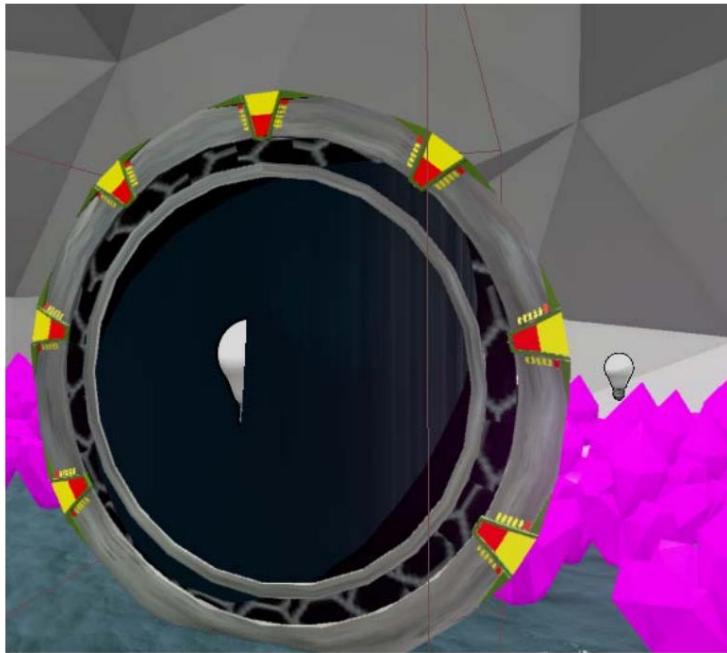


图 61. 在游戏结束时从编辑器中捕获门。

来源:自制。

5.1.4.8.封闭材料

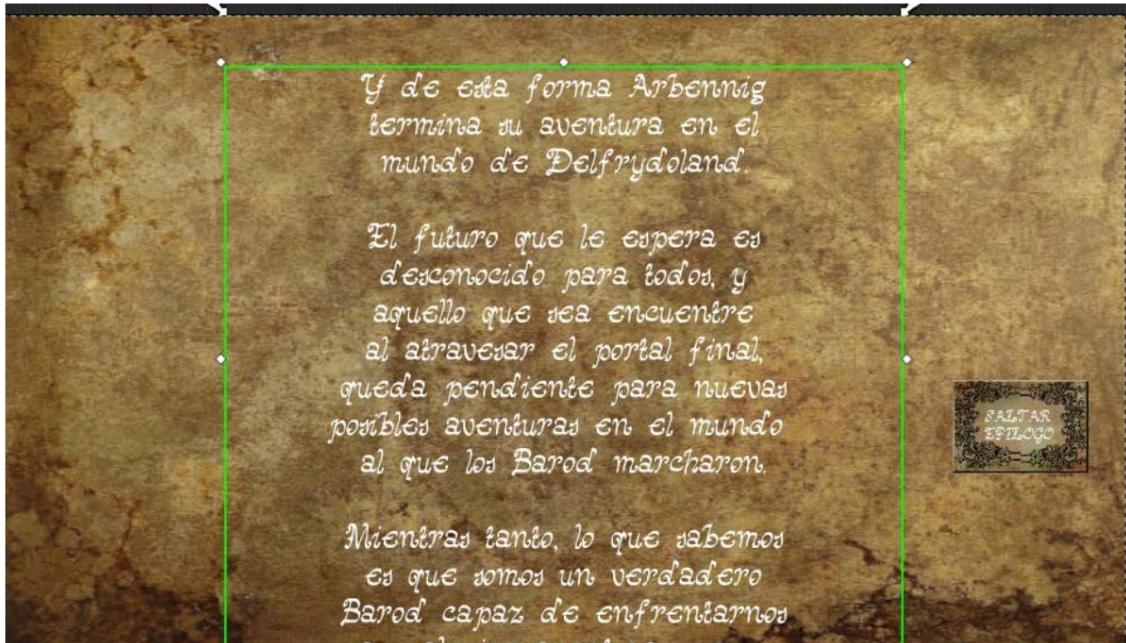


图 62.从编辑器中捕获的尾声。

来源:自制。

当玩家触摸传送器或与传送器交互时,会加载一个带有文本的屏幕

动画,包含游戏的结尾。尾声告诉打过的玩家

达到水平,您正在冒险进入新世界。

结语结束时,如果您不想看,可以跳过,您转到

学分,遵循结语和介绍的相同操作。最后到

完成学分或跳过它们,您可以访问主菜单。

5.1.5.界面

5.1.5.1.平视显示器

HUD (平视显示器) ,请记住关于

最先进的,它是出现在游戏屏幕上的 2D 信息,显示

对玩家有用的信息。



图 63. 编辑器中游戏关卡中永久 HUD 的屏幕截图。

来源:自制。

在 Arbannig,一旦游戏开始,我们就会在 HUD 上永久显示:

- 角色的生活。
- 角色的能量。
- 获得和未获得的符文。

另一方面,HUD 会在玩家激活它时显示游戏库存,反过来

反过来,当我们从清单中选择一个对象时,它会出现一个操作子菜单。

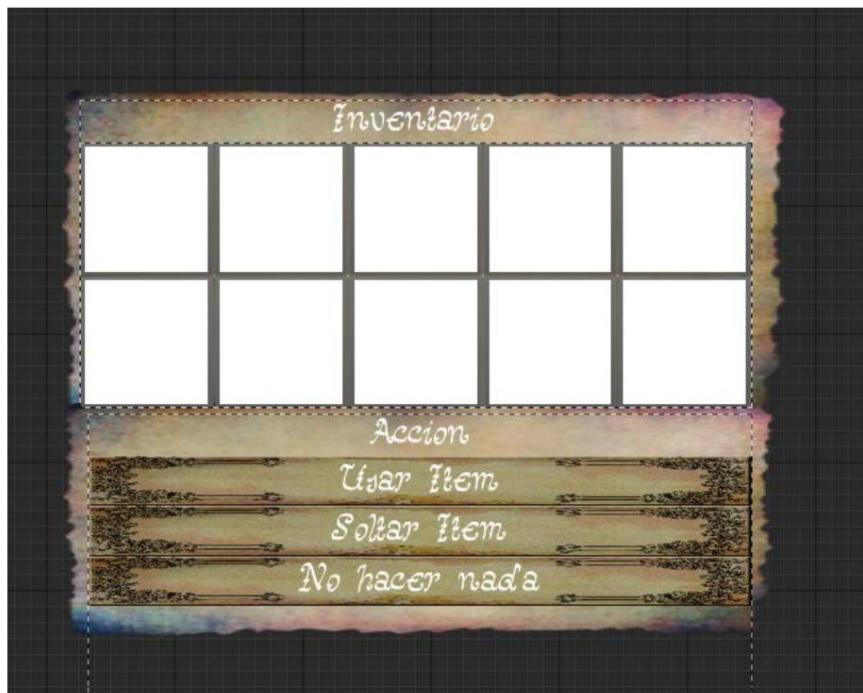


图 64. 从编辑器中捕获库存。

来源:自制。

我们还会显示有关收集的戒指和剩余时间的信息。

环挑战测试。



图 65. 从编辑器中捕获计时器和循环计数器。

来源:自制。

无论玩家是否通过了任何图腾挑战或保存了游戏,或者他们是否可以通过 HUD 使用任何物品,我们都会弹出通知。



图 66.来自 HUD 的其他通知。

来源:自制。

5.1.5.2.菜单

菜单将允许我们访问游戏关卡、修改游戏的各个方面（分辨率和一般音量），访问额外关卡，退出游戏，暂停游戏或访问积分，除其他事项外。

接下来，我们将解释这些菜单和屏幕中的每一个，我们将为此支持游戏流程摘要部分（5.1.1.5）中的游戏流程图。

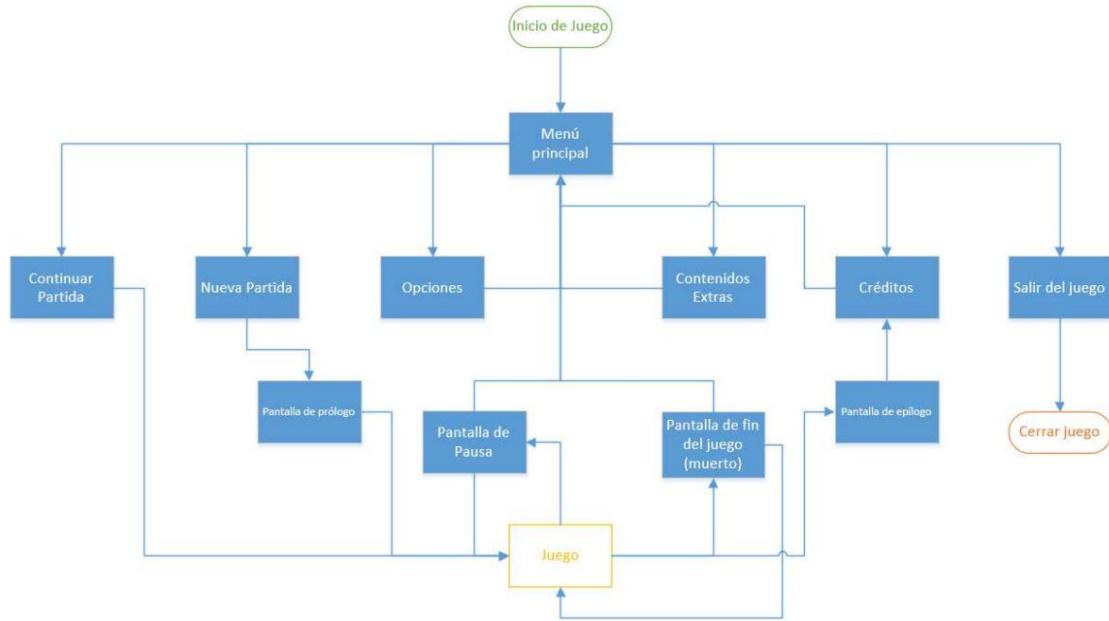


图 67. 游戏菜单图。

来源：自制。

5.1.5.2.1。主菜单。



图 68. 编辑器主菜单的屏幕截图。

来源:自制

此菜单是您开始游戏时打开的菜单。在其中我们找到启动或加载的选项

(如果我们保存了数据)游戏,以及对选项菜单的访问,在级别

学分和额外费用。正是通过这个菜单,我们可以退出游戏,从选项

离开。

5.1.5.2.2. 选项菜单。



图 69. 编辑器中选项菜单的屏幕截图。

来源:自制

它允许我们修改屏幕分辨率,让我们可以选择三种 4:3 格式,即 640x480、800x600 和 1024x768,以及其他三种 16:9 格式,即 1280x720 (默认显示的选项)、1366x768 和 1920x1080。设置分辨率时,按钮

选择它被禁用。

此菜单还允许您通过滑块修改游戏的总体音量,并通过

最后,返回主菜单。

5.1.5.2.3。暂停菜单。



图 70. 从编辑器中捕获暂停菜单。

来源:自制

当玩家暂停游戏 (显然)时出现的暂停菜单让我们可以选择
返回游戏或返回主菜单。

5.1.5.2.4。新游戏确认菜单



图 71. 编辑器中新游戏确认菜单的屏幕截图。

来源:自制

页。 76

当我们点击新游戏主菜单的选项时,这个菜单会跳转,现有
一个现有的已保存游戏。从此菜单中,我们可以返回主菜单,或者确认我们希望删除游戏并开始新游戏。

5.1.5.2.5.游戏结束菜单（当玩家死亡时）



图 72. 玩家死亡后的游戏结束菜单。

来源:自制

当玩家接触到死亡平台（游戏中的海水）,或与各种破坏平台发生碰撞并耗尽生命值时,会加载此菜
单。

在这里,我们可以选择从最后一个保存点开始,或者返回到主菜单。

序幕画面。

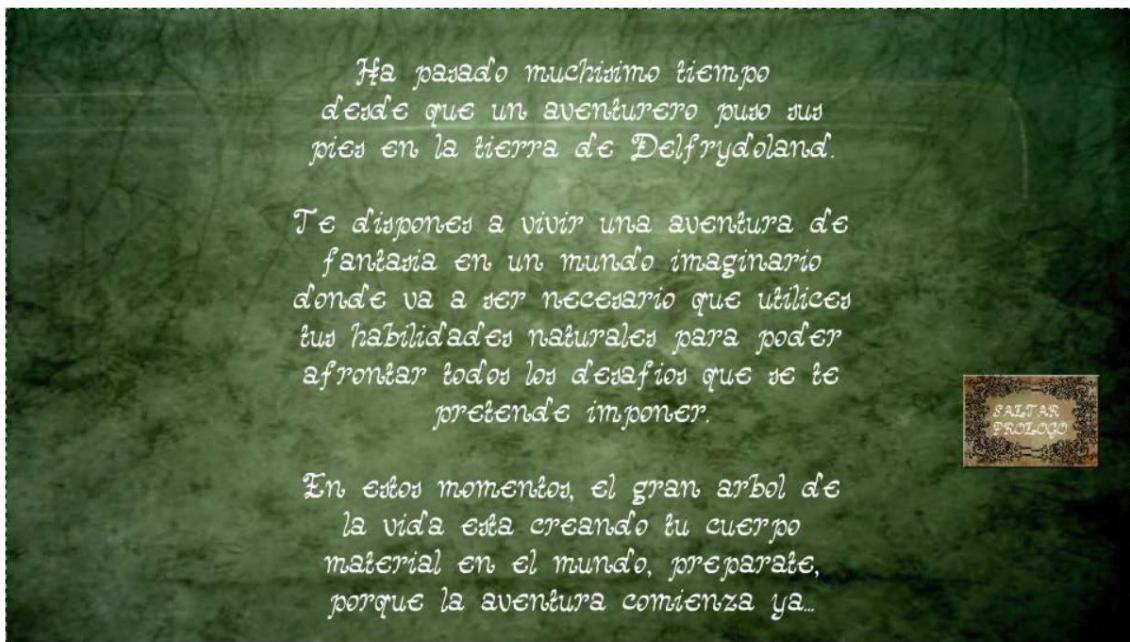


图 73. 编辑器的序幕屏幕截图。

来源:自制

该屏幕本身也用作菜单。它是一个带有动画文本的临时屏幕,当文本上升时,游戏加载。

玩。它还提供了通过按钮跳过序幕的可能性。

尾声画面。

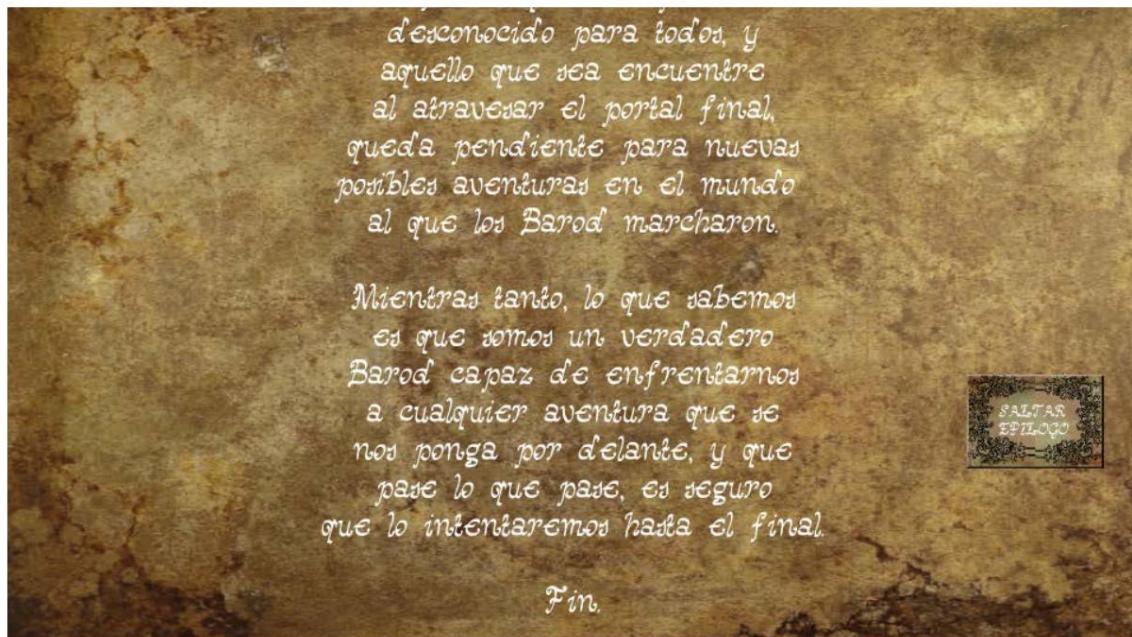


图 74. 编辑器的序幕屏幕截图。

来源:自制

页。 78

其操作与序幕相同,不同之处在于
通过这个屏幕,我们访问的不是游戏,而是游戏积分。
学分屏幕。



图 75. 编辑器的序幕屏幕截图。

来源:自制

与序言和尾声一样,演员表的工作方式相同,但它们指向主菜单。

5.1.5.3. 相机

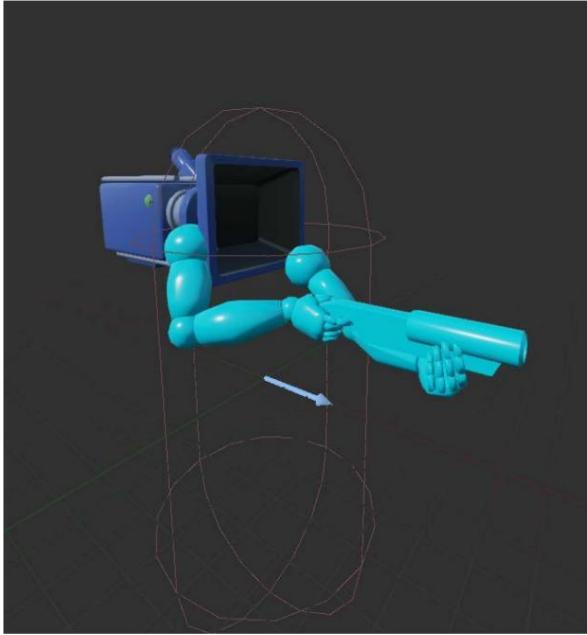


图 76. 从显示其组件的编辑器中捕获的字符。

来源:自制。

Arbannig 的相机是第一人称相机。对应愿景

游戏中的角色,从他的角度代表世界。

此外,它是我们在项目中唯一拥有的相机,因为其余的组件

它们是在屏幕上以 2D 形式显示的菜单和动画。

5.1.6. 声音

视频游戏中的音频旨在伴随菜单、过场动画和游戏关卡。

为方便起见,我们将其分为音乐、音效和 UI (用户界面),所以用音乐

我们指的是更像是不断播放的背景歌曲的音频,以及 SFX 和

UI 是音效,尽管 SFX 可以是玩家动作 (它们听起来像

UI 通过在世界上没有位置)或没有 (它们听起来在世界上的某个位置,以及多少

我们离他们越近,他们的声音就越大)。

关于音乐,取决于我们是否在菜单、电影、级别

游戏或附加内容,一首或另一首歌曲将在背景中循环播放。

如前所述,SFX 和 UI 将伴随世界的不同组成部分,

角色,以及玩家通过菜单的动作 (按下按钮时)

或悬停在上面)。

5.1.7.帮助系统

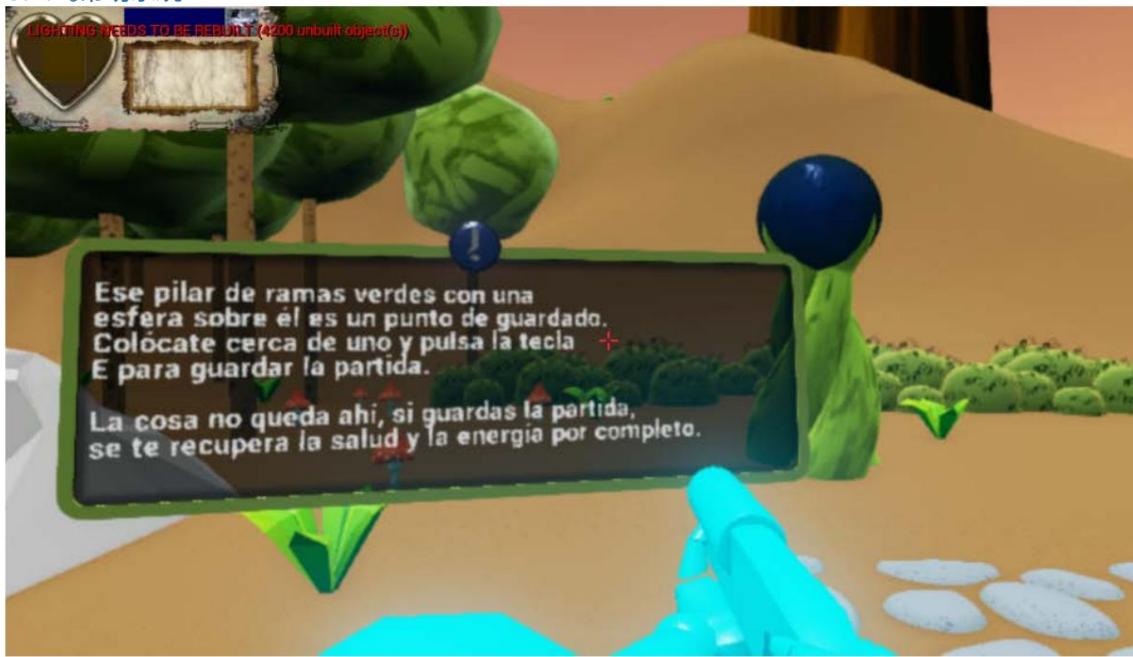


图 77. 信息点捕获已激活。

来源:自制。

为了协助玩家开始游戏,起始岛包含一系列
信息性帮助消息,显示为关闭,并在播放器时显示
离他们足够近。这些消息为玩家提供了关于
游戏控制,以及一些玩技巧。



图 78. 未激活的信息点捕获。

来源：自制。

5.1.8。人工智能

5.1.8.1。 NPC AI（生命之树、老年村民和最终岛图腾）

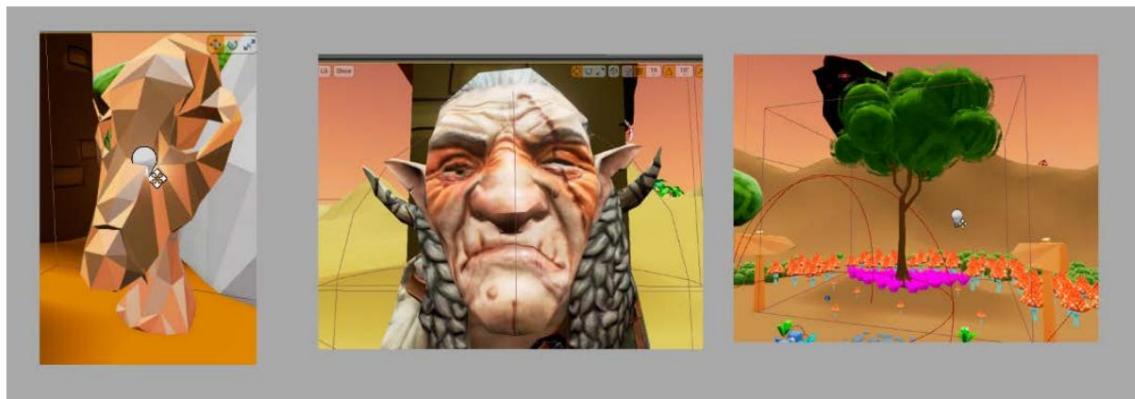


图 79。图腾，看起来像老人的村民，以及编辑器的生命之树。

来源：自制。

关于生命之树、老样子的村民、最后的海岛图腾,他唯一的
游戏逻辑包括能够与他们开始对话的可能性。

当玩家在他们附近互动时,会打开一个对话窗口,显示 NPC 的消息以及来自玩家的一个或多个响应。

这些角色不执行任何其他类型的动作,他们总是静止不动,不巡逻。

5.1.8.2.符文守护者图腾的AI。

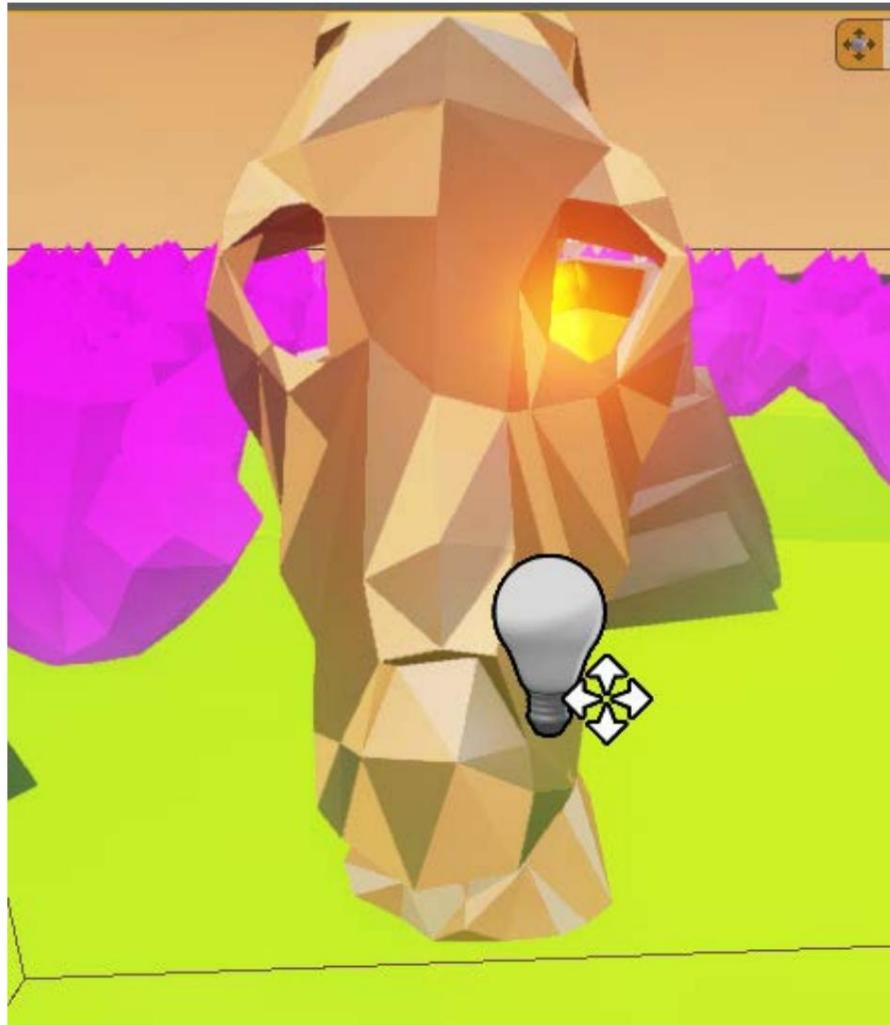


图 80. 从编辑器中捕获的图腾。

来源:自制。

符文守卫图腾具有与 NPC 几乎完全相同的功能

正常的,另外通过与他们交谈,可以接受他们准备好的挑战 (如果他们有挑战的话),并且一旦完成,他们就会给我们他们保留的符文。所以,在这之后,我们就不能和他们说话了,因为他们的灵魂离开了他们的身体

交付了符文。

5.1.8.3.魔像人工智能。



图 81. 从编辑器中捕获傀儡。

来源：自制。

出现在第六个符文挑战中的石傀儡的工作原理如下：

1. 站着不动，看不到玩家。
2. 当他看到玩家时，他会在地图上追逐他。
3. 当它到达它并与它碰撞时，它又变得静止了。

它的操作是这样的，因为符文6的挑战条件是
玩家必须跑到岛尽头的一面旗帜，然后，

带她到图腾指示的地方。路上到处都是傀儡，他们一看到玩家，他们会追逐它，如果他们碰到它，玩家就会失去挑战，必须再试一次。

5.1.9。技术指南[25]

关于游戏的硬件、软件和架构要求的一些技术细节如下所述。

5.1.9.1。硬件

关于玩游戏所需的硬件，建议至少有以下几点：

内存 RAM :8 GB

处理器：英特尔或 AMD 四核类型

显卡：兼容 DirectX11。

5.1.9.2。软件

推荐使用 64 位架构的 Windows 7 或 8。

5.1.9.3。游戏架构

游戏遵循的架构与框架部分中已经解释的架构相对应。

理论或现有技术 (2.2.4)。也就是虚幻引擎所基于的架构

4.

5.2.开发和实施。

我们准备解释制作我们设计的视频游戏的过程
之前在 GDD 中 , 我们继续解释开发的各个方面 ,
或多或少地强调它们取决于这些最相关的方面
包含。

5.2.1.项目创建

所有步骤中的第一个是项目的创建及其相应的设置。

在我们的案例中 , 鉴于它与 Oculus Rift DK1 的兼容性 , 我们通过
Epic Games Launcher 是一个 4.6.1 版的空白蓝图项目 , 自
以后的版本已经亲自验证 , 这不能正常工作
Oculus DK 版本。 创建后 , 我们可以从
发射器。

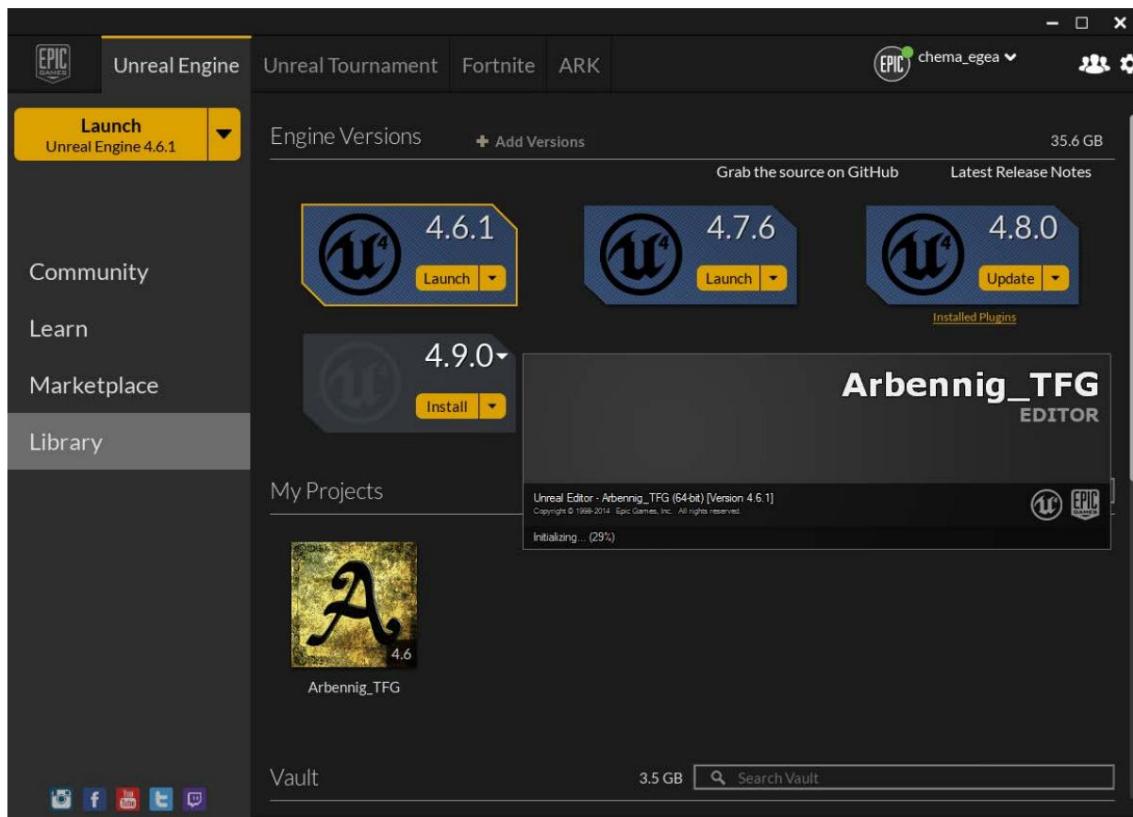


图 82. Epic Games 启动器的游戏启动屏幕截图。

来源 : 自制。

创建项目并访问它后,有必要对其进行配置,我们从编辑我们稍后需要设置项目的类。

我们要创建的基本类是:

- 一个 GameMode 类,我们将在其中建立游戏规则。
- 一个HUD类型的类（它不同于我们将有视觉的HUD,因为这个类 HUD 与代码一起工作,是我们渲染目标的地方）。
- 一个将帮助玩家拥有角色的 PlayerController 类 主要的。
- 一个 Character 类,它将具有 PlayerController (虽然不是默认情况下,因此只有在游戏期间我们控制角色,同时,我们有一个不响应玩家事件的自由视图,仅对那些我们用鼠标指示菜单)。我们创建 Character 而不是基本 Pawn,因为使用 Character 我们已经拥有人形角色功能将促进我们的发展。
- 一个 PlayerCameraManager 类,我们将分配给 PlayerController 以拥有控制我们创建的相机。

一旦创建了这些类,我们就可以访问编辑器的项目设置,然后修改我们认为必要的价值观。在我们的例子中,我们已经修改 (我们必须考虑到请注意,我们已经创建了级别和角色,在本实施指南中已定义后面的实现):

描述,添加创建者、项目数据的相关信息。

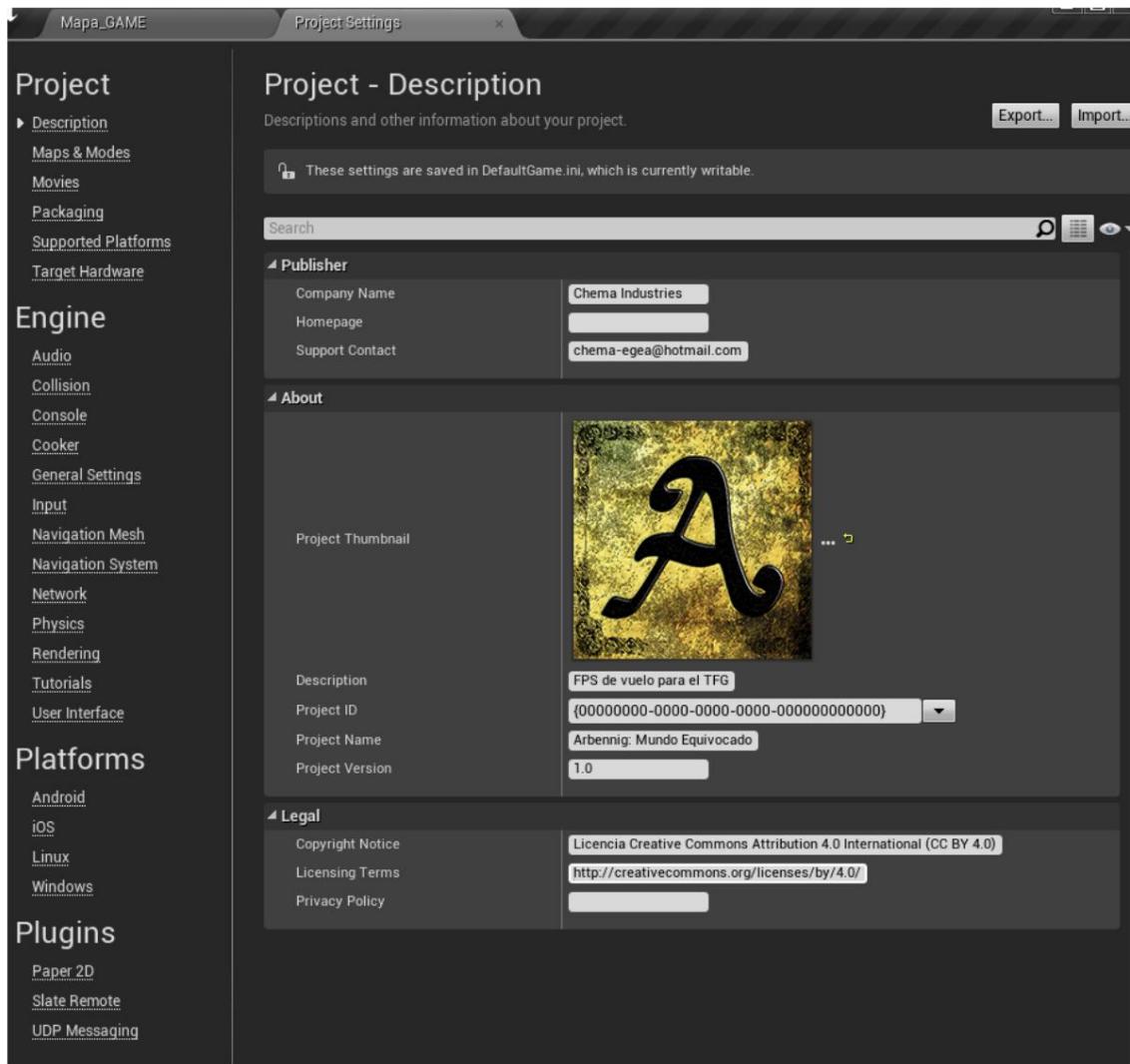


图 83. 捕获在 Arbannig 的描述部分中添加的修改。

来源:自制

Maps & Modes,选择加载游戏 (游戏地图)和编辑器 (主菜单地图)时默认加载的地图,以及游戏模式,在理论框架或最先进部分 (2.2 .4)中描述,我们在其中指出我们的 GameMode 类,如果我们有一些默认的 Pawn 类 (在我们的例子中,我们没有分配,而在游戏中我们有一个 Character),我们创建的 HUD,PlayerController 类,最后是该类游戏状态。

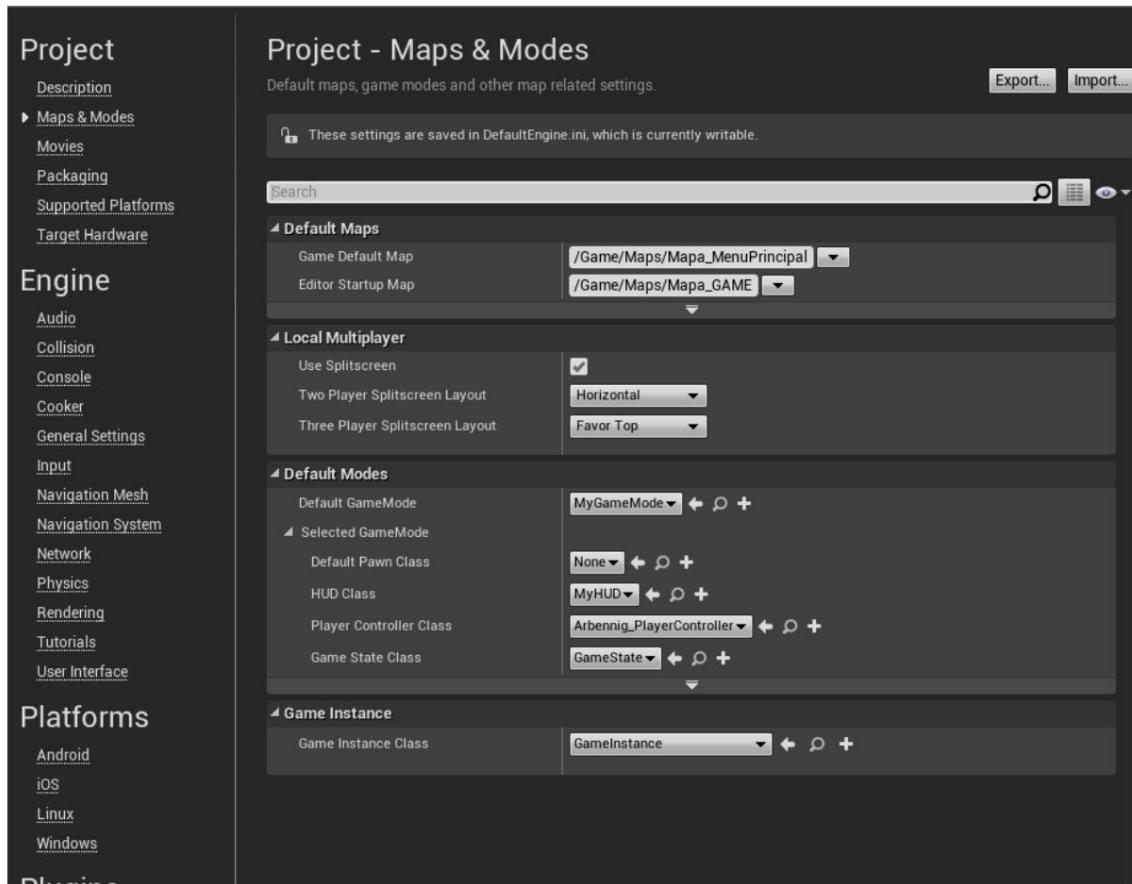


图 84. 捕获 Arbannig 项目的 Maps & Nodes 配置。

来源:自制。

支持的平台,选择我们只想为 Windows 制作游戏。

音频,我们选择要使用的默认声音类,在我们的例子中,我们有
创建了类 BP_Master_Sound (5.2.4)。

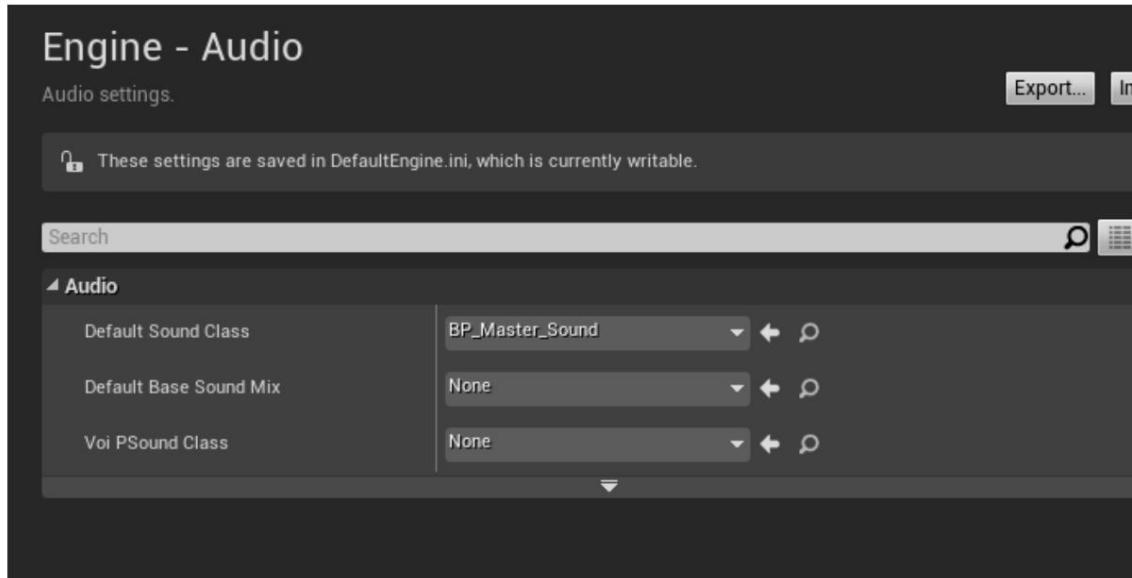


图 85. 指示在何处选择游戏声音类的屏幕截图。

来源:自制。

输入,其中为游戏控件添加了标识事件,也就是说,它已经添加了控件的定义列表,以便我们以后可以访问它们来自蓝图的图表编辑器,按我们指定的名称。已决定纳入此为 Xbox 控件留出控件,预计未来可能的改进以允许其使用。

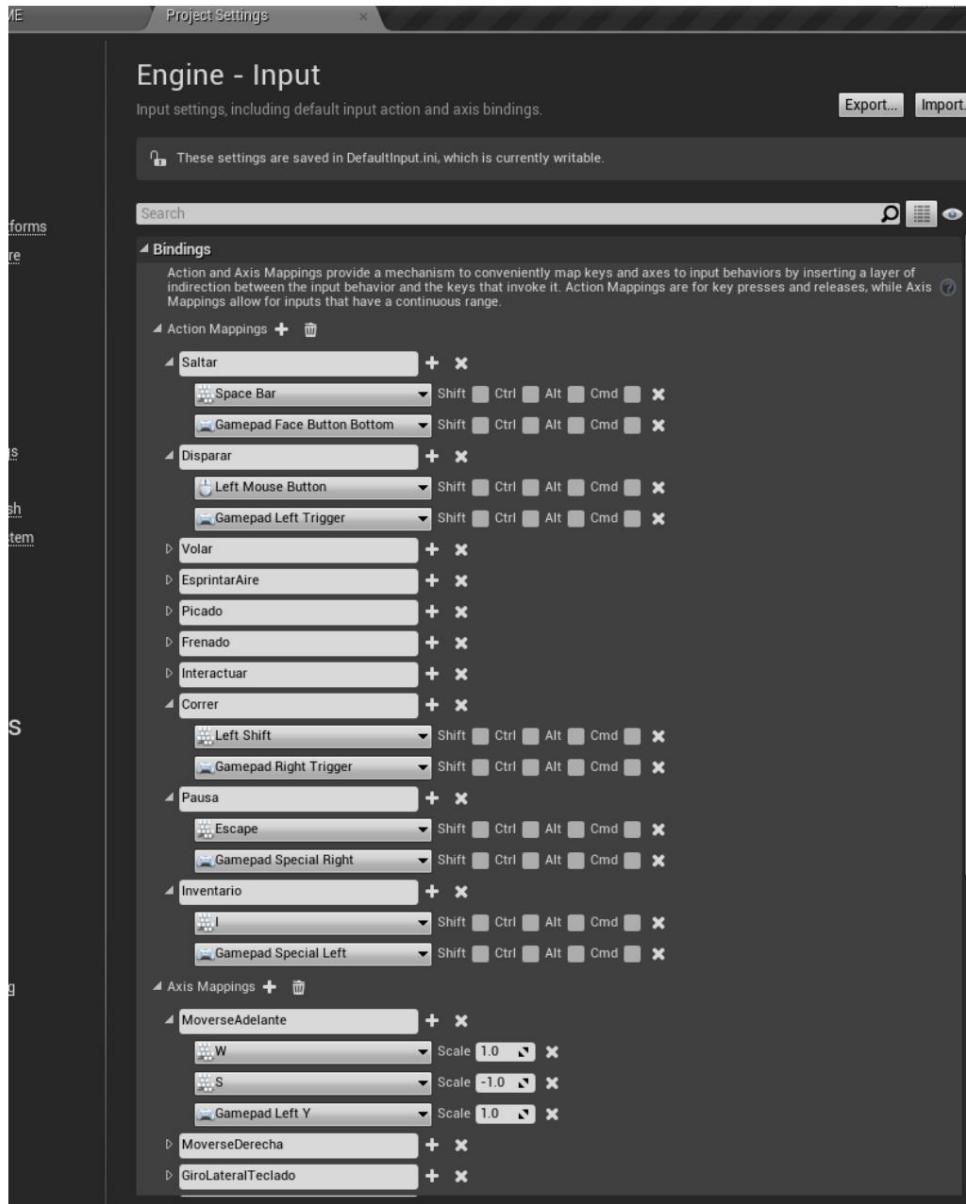


图 86. 捕获项目的 Inputs 配置。

来源:自制。

值得一提的是输入部分,我们有动作映射值,并且轴映射值。这些之间的区别在于,您可以使用 Axis Mappings 保存也是一个浮点类型的值,因为它们是用于移动的,如果我们想要的话,这样更容易保存只要玩家按下一个键,例如,上升,我们就简单地去将浮动添加到玩家的位置。

另外,最后要注意的是,对于轴映射的值,不方便为每个角色的动作做出定义,但我们可以只需将您的浮动设置为负数。例如,如果角色向前移动 W,这有一个刻度为 1.0 的浮点数,我们添加到这个向前移动的动作,如果你按下 S 键,它的浮点数将为 -1.0,这样,我们已经定义了控制正面和背面。



图 87. 运动控制定义向前和向后捕获。

来源:自制。

还需要为我们的 PlayerController 分配一个 PlayerCameraManager 类,它允许我们让您可以控制游戏摄像机。

只需在 PlayerController 定义中,我们选择创建的类。

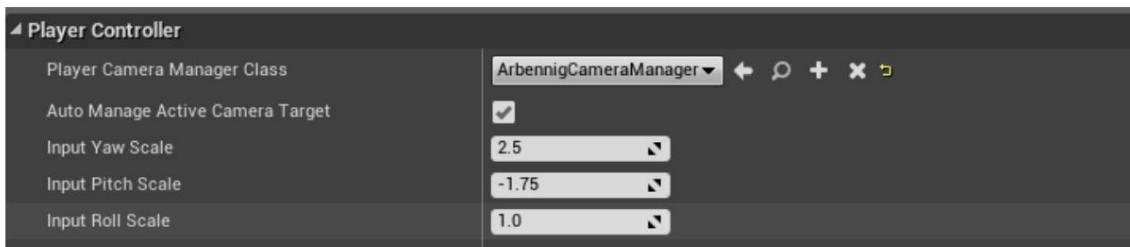
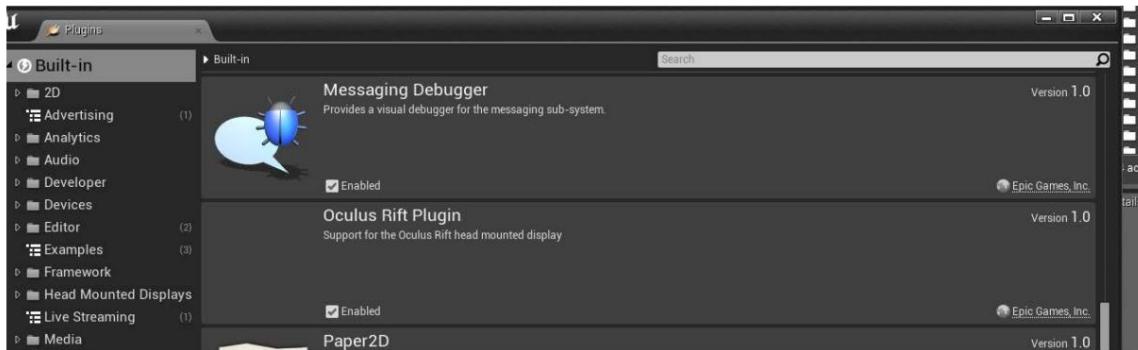


图 88. 分配给 PlayerController 的已创建 PlayerCameraManager 类的捕获。

来源:自制。

最后,我们在 Window 中找到的插件窗口中,我们激活 oculus 插件 rift,默认情况下包含在引擎中。



5.2.2. 执行角色的动作。

接下来,我们继续解释创建运动控制的蓝图

角色基于他们的状态变化（地面,飞行和赛车）。控制蓝图

我们选择在 Character 类中进行,在我们的例子中是 Arbennig_Character,就是这样

类可能是整个游戏中最广泛的,还有关卡蓝图,它是如此广泛

通过从图形编辑器缩小,我们只能看到一小部分

这个蓝图,所以决定解释最重要的步骤

与其创建相关,但不详细,因为这

内存几乎翻倍。

也就是说,我们应该知道,由于 Character 类提供了一个枚举

不同的状态,包括Walking（我们将用于地面的状态）,Flying（

声明我们将用于飞行)和 Falling（我们将在玩家决定执行

直线下降）。但为了帮助我们进行开发,我们还将创建几个

列出,一种用于地面状态,其中将包含行走和坠落,另一种用于

air,它将包含 Flying 和 Racing（我们将这些列为变量添加到

字符以便能够使用它们）。这些状态的变化流程如下：

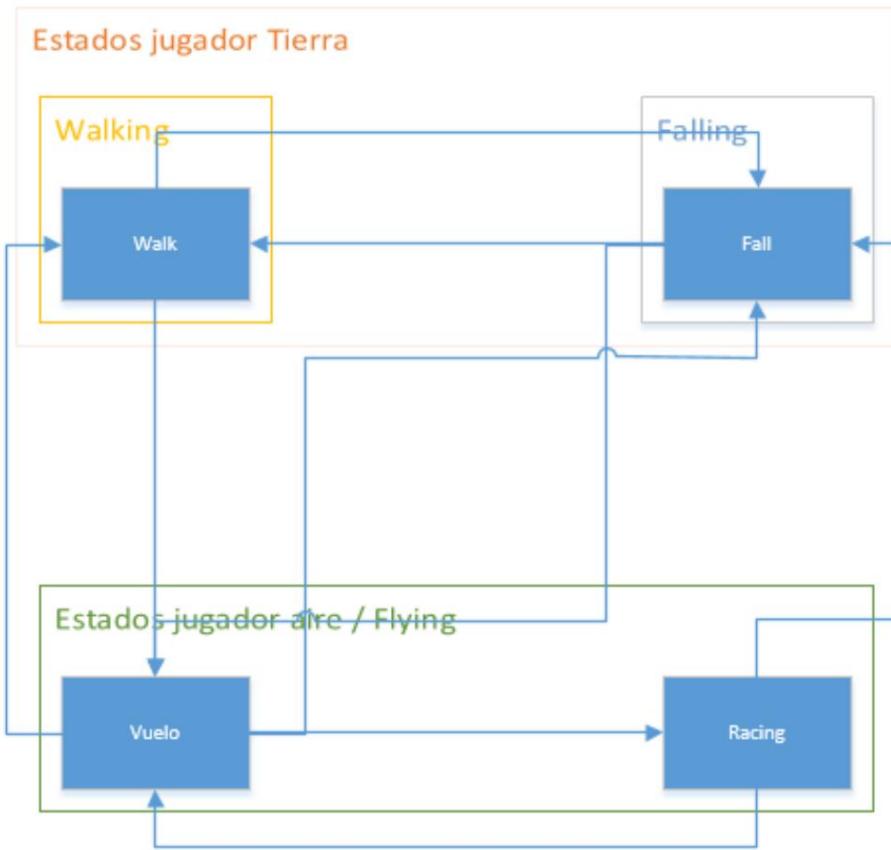


图 89. 状态变化图。

来源:自制。

如您所见:

- 从 Walk 我们可以去 Flight 和 Fall 状态。
- 从 Flight 我们可以去 Walk、Fall 和 Racing 状态。
- 从赛车我们可以去秋季和飞行。
- 从秋季开始,我们可以进入 Walk 状态和 Flight 状态。

有必要知道,这样我们才能拥有类的不同状态的功能

Character,我们要做的就是在它的初始化参数中,勾选 Can
飞行,可以跳跃,y 可以行走,en su 运动组件。



图 90. 捕获允许玩家在默认移动状态之间切换的变量。

来源:自制。

5.2.1.1. 相机旋转。

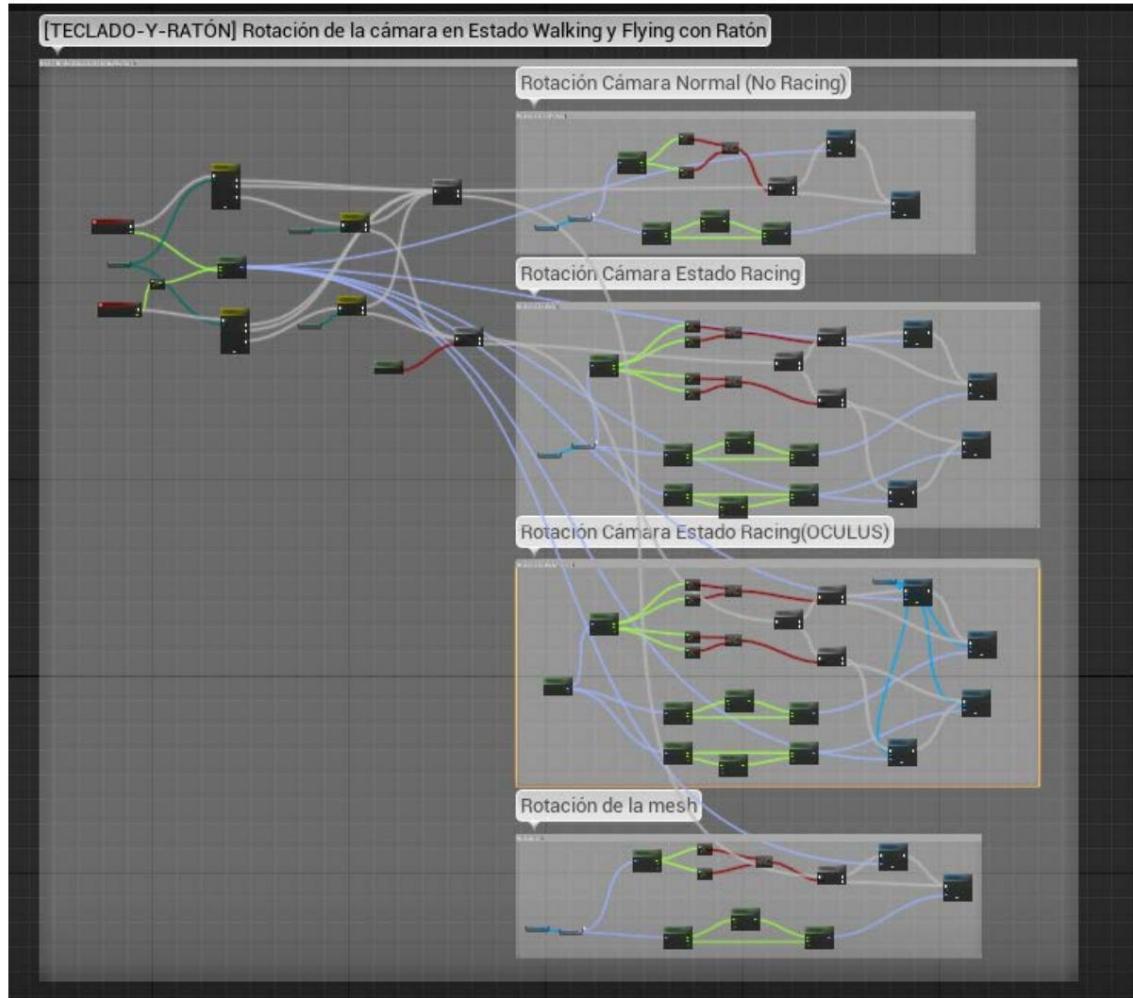


图 91. 摄像机旋转图部分概览。

资料来源:自己的阐述。

玩家的轮换分为两部分,一方面是镜头,另一方面是镜头。

另一个,它具有的网格。这是由赛车控制决定的,您可以在其中移动
自由相对于身体。

对于相机旋转的实现,我们首先做一系列的检查首字母来确定玩家的运动状态,以及他们是否在眼中连接的。此外,我们使用鼠标输入创建一个旋转器(旋转向量),这样我们就可以稍后将它们添加到相机和网格旋转中。

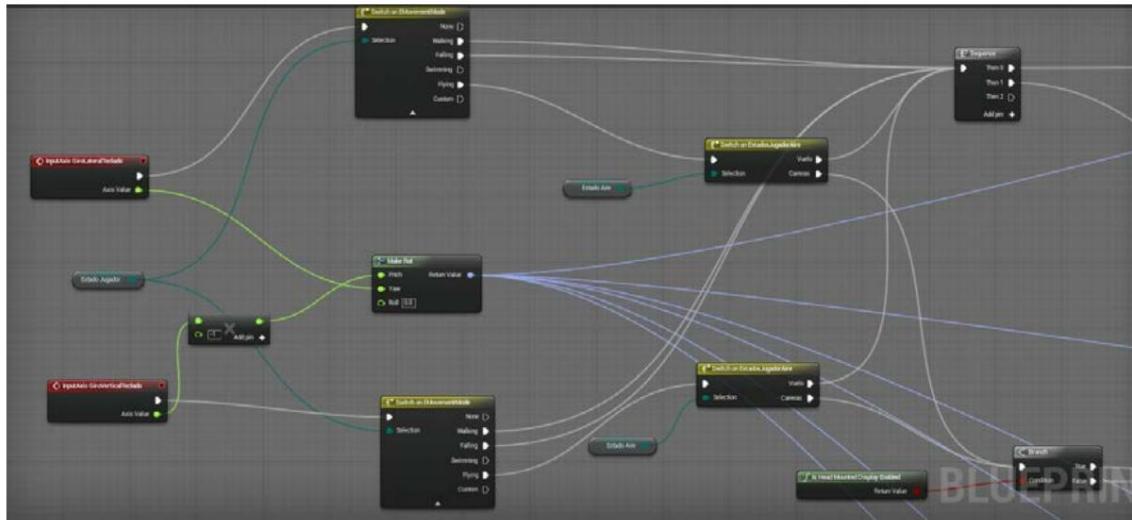


图 92. 在继续旋转之前捕获初始检查图表的一部分。

来源:自制。

在此之后,如果我们不在赛车模式下,则应用摄像机旋转时会考虑我们所处的角度的钳位,以免超过一定量,并且在尝试超过 90° 限制时会出现旋转问题。最后,如果我们不是,则添加旋转

超过任何限制,然后再次设置旋转以避免由于尝试同时应用俯仰和偏航旋转而在虚幻中出现的相机崩溃问题,我们通过这种方式修复了它。

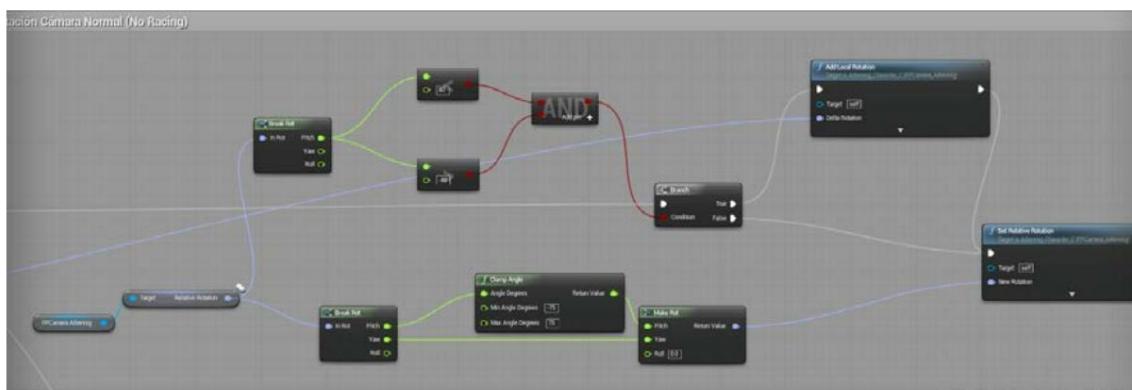


图 93. 不处于赛车模式时的摄像机旋转捕获。

来源:自制

至于 Racing 状态下相机的旋转,有点奇特,因为它是设计为与 oculus 兼容是正确的。当我们进入赛车模式时,相机与玩家的身体分离,是自由的,因此我们可以看到在特定方向飞行时的环境。尽管如此,与相机的区别不是赛车的是,这里我们限制了俯仰角(上/下)和偏航角(向两侧),因为角色的头部不能向后转动。

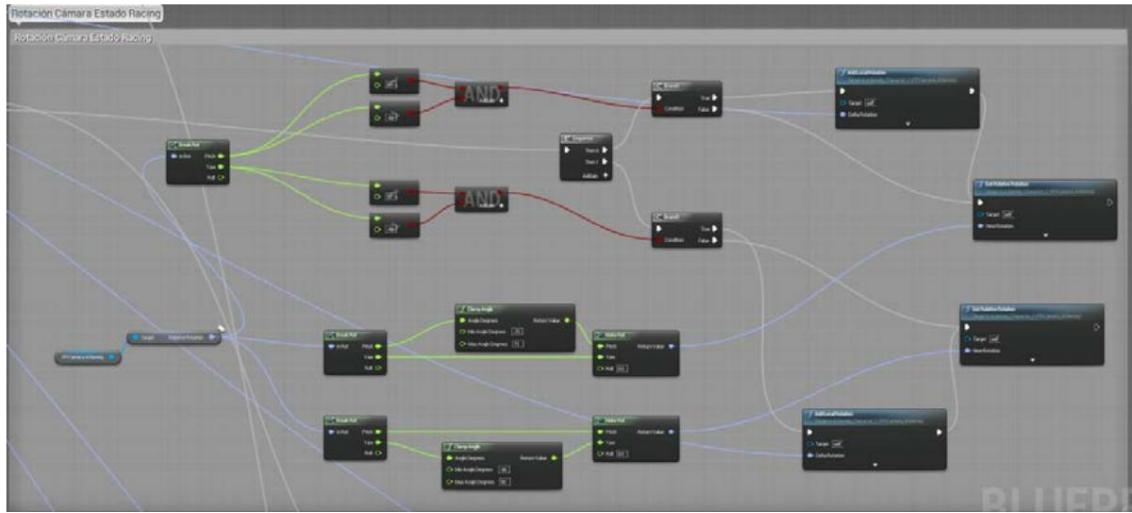


图 94. 赛车模式下的摄像机旋转捕捉

来源:自制

对于赛车中的 oculus 相机旋转,主要区别在于我们得到从外围设备而不是直接从我们拥有的相机进行定向,其余的步骤是一样的。

图 95.当我们处于赛车模式并激活 oculus 时的摄像机旋转捕获。

页。 96

来源:自制

最后,关于角色网格的旋转,它会随着相机的旋转而旋转

当我们处于赛车以外的状态时。因为在赛车中它会随着角色本身转动

(请记住,角色是一组网格和相机,所以我们有一个角色实体,它可以旋转和移动它的所有组件,另一方面,我们可以分别访问其每个元素并进行必要的修改)。

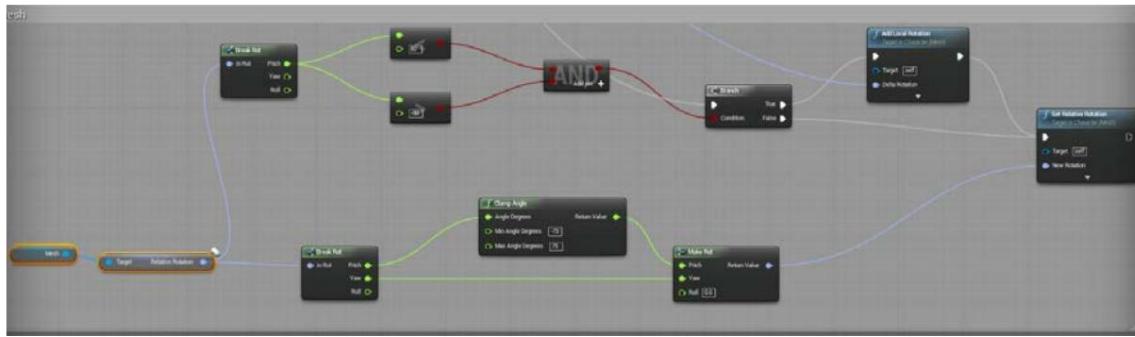


图 96. 网格旋转代码部分的捕获。

来源:自制

5.2.1.2。人物运动。

角色的动作取决于我们所处的状态,
因为 Racing 状态的操作逻辑与
其余州,在其余状态下使用 WASD 键移动,而在 Racing
有一个外部推动我们前进,而使用 WASD,我们所做的就是改变角色。

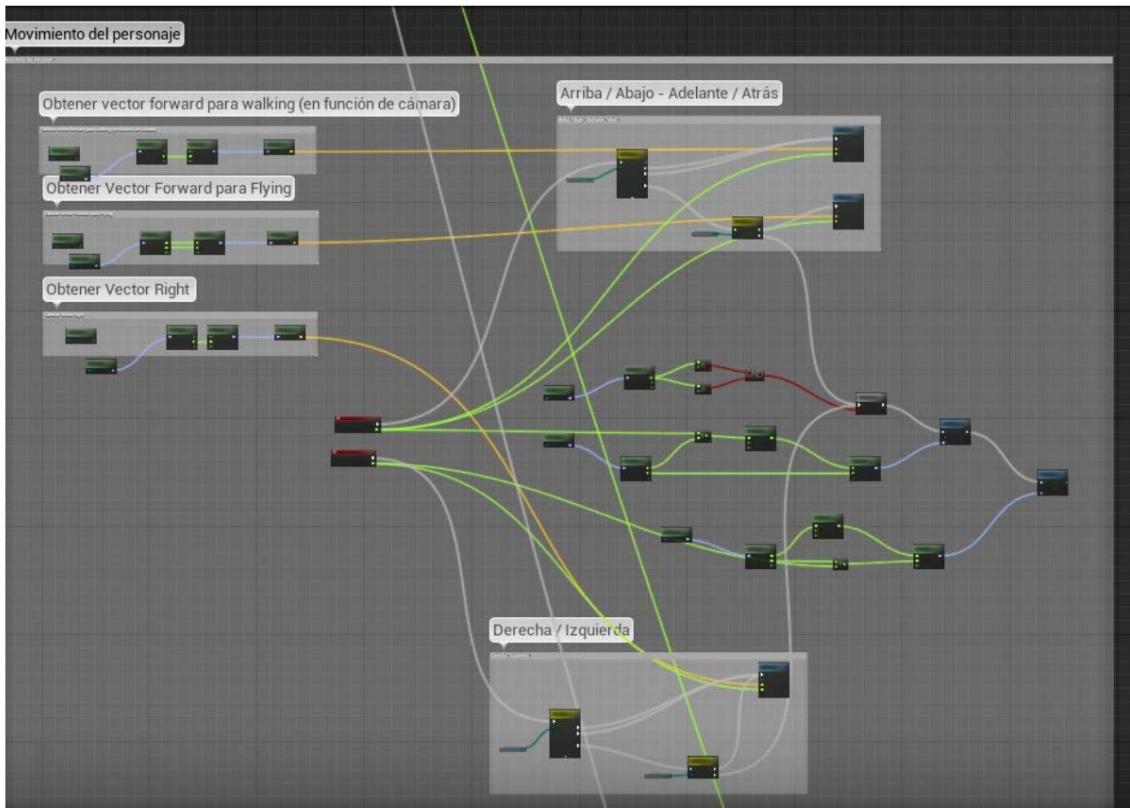


图 97. 角色在所有状态下的运动的一般捕获。

来源:自制

只要我们不在赛车模式下,我们要做的第一件事就是获得它的 Forward (用于向前和向后移动)和 Right (用于横向移动)向量。这些向量可以作为旋转的函数获得

角色自动出现在世界中,因为引擎为我们提供了一个功能

它你只需要通过一个旋转器。

因为在行走中,我们只能在陆地上移动,也就是说,在空间中

“二维”,在飞行中我们也可以自由上下移动。

下,我们计算两个不同的前向向量。

对于右向量的情况,我们不需要这种区分,因为它是相同的。

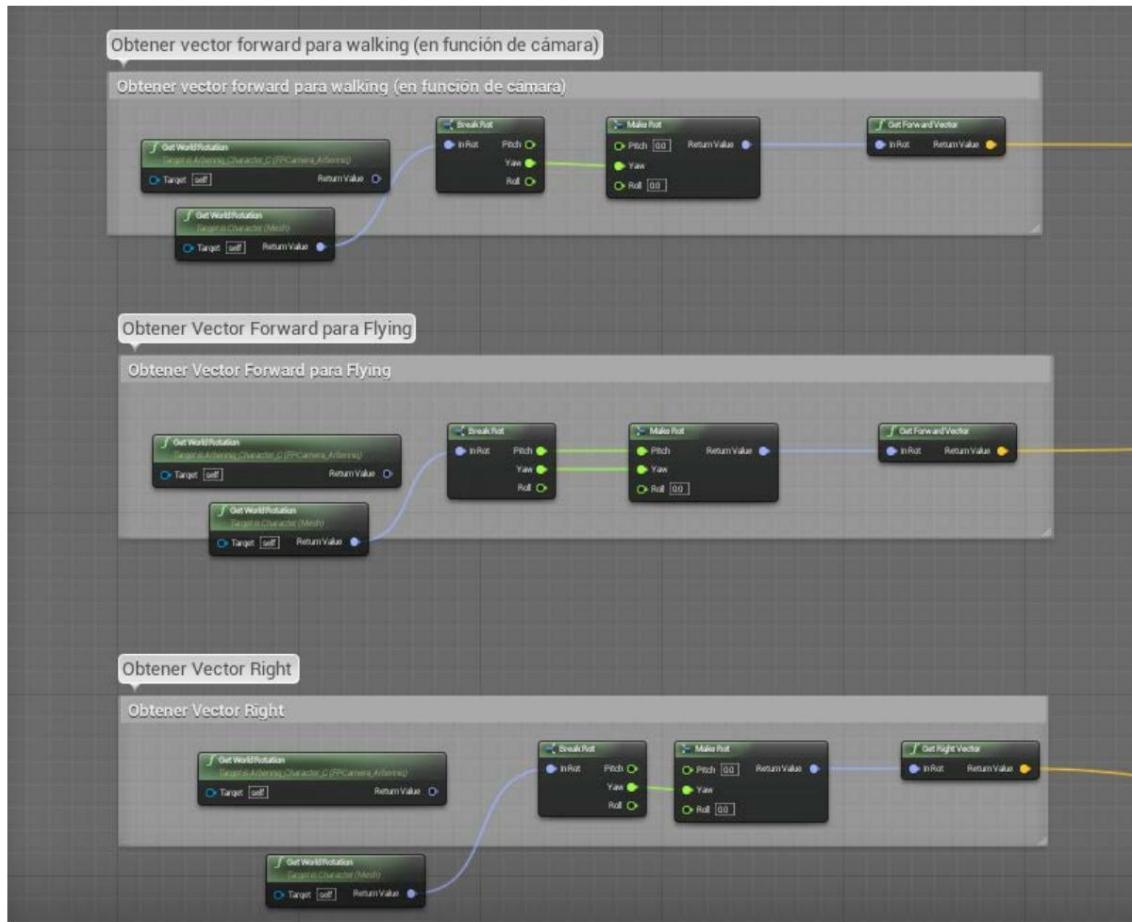


图 98. 获取指向前方和右侧以确定运动的矢量的捕获。

来源:自制

获得向量后,只需将运动添加到我们的 Pawn 中

这些向量的方向,既用于向前/向后移动,也用于横向运动。

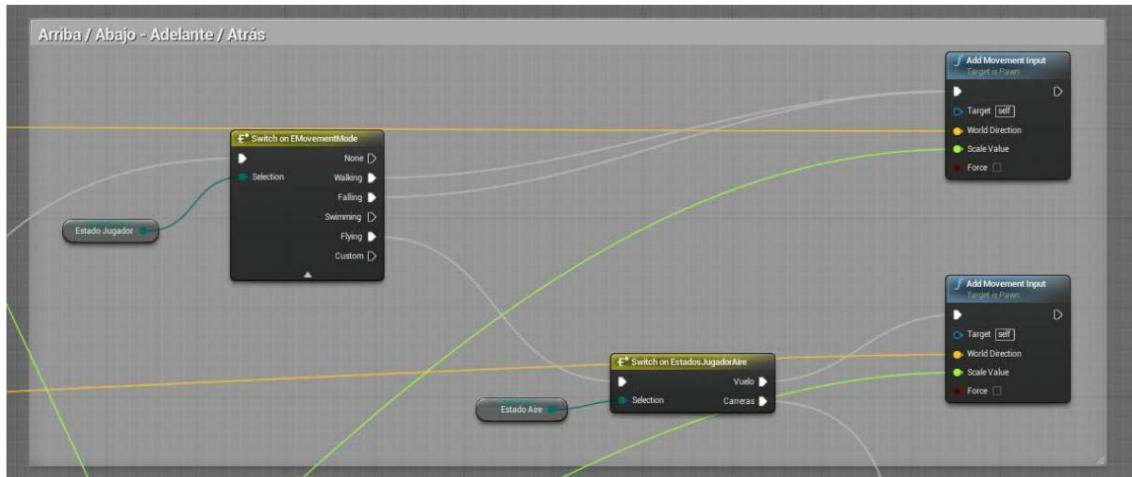


图 99. 捕获步行、坠落和后退运动逻辑

飞行（飞行）。

来源：自制

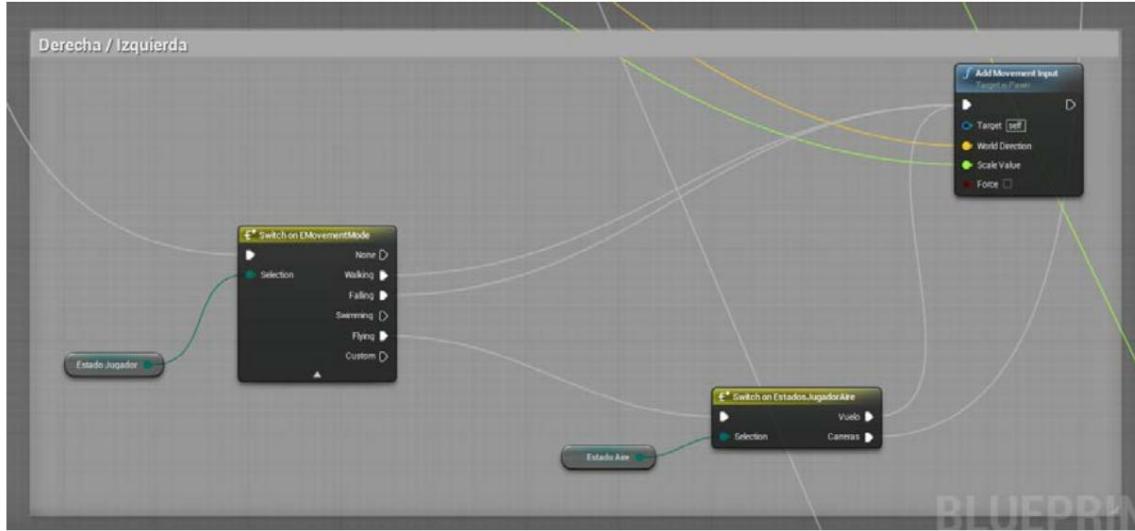


图 100. 捕获移动逻辑以在其中向右和向左移动

行走、坠落和飞行。

来源：自制

另一方面,对于处于 Racing 状态的运动,如上所述,它已经在进入所述状态时不断应用,因此当玩家按下任何在 WASD 键中,它所做的是转动 Actor (建立,是的,就像转动如上所述,一些角度控制,使它们不超过一定的量,给麻烦)。

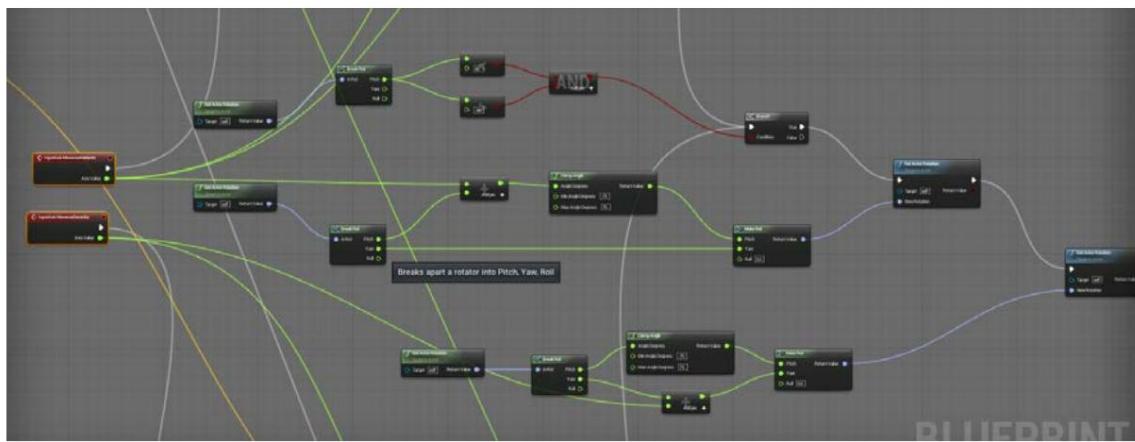


图 101. Racing 中运动的屏幕截图。

来源:自制

5.2.1.3。上升/下降

应用了仅在飞行状态下发生的角色的垂直上升和下降
只需使用其 Ascent 事件将 Z 轴上的运动添加到我们的角色。

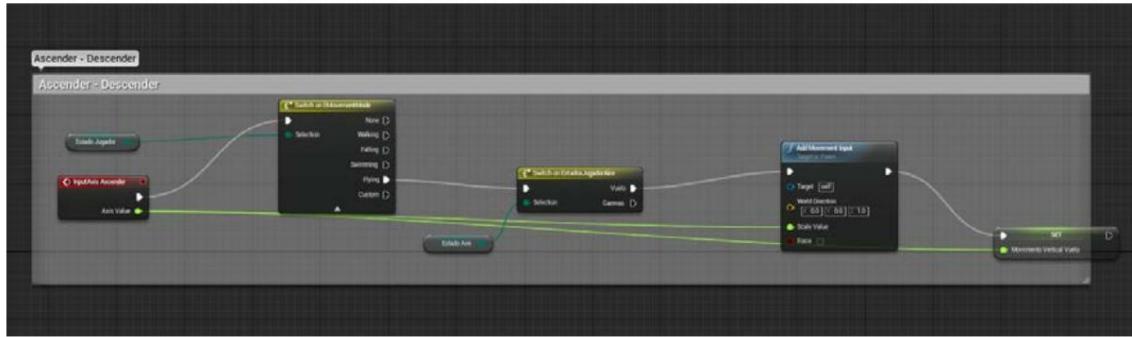


图 102. 捕捉飞行中上升和下降的逻辑（不是赛车）。

来源:自制

5.2.1.4。跳过

跳转逻辑由已经实现的字符类提供,但要将其应用到我们的
字符,我们控制它所处的状态是行走。

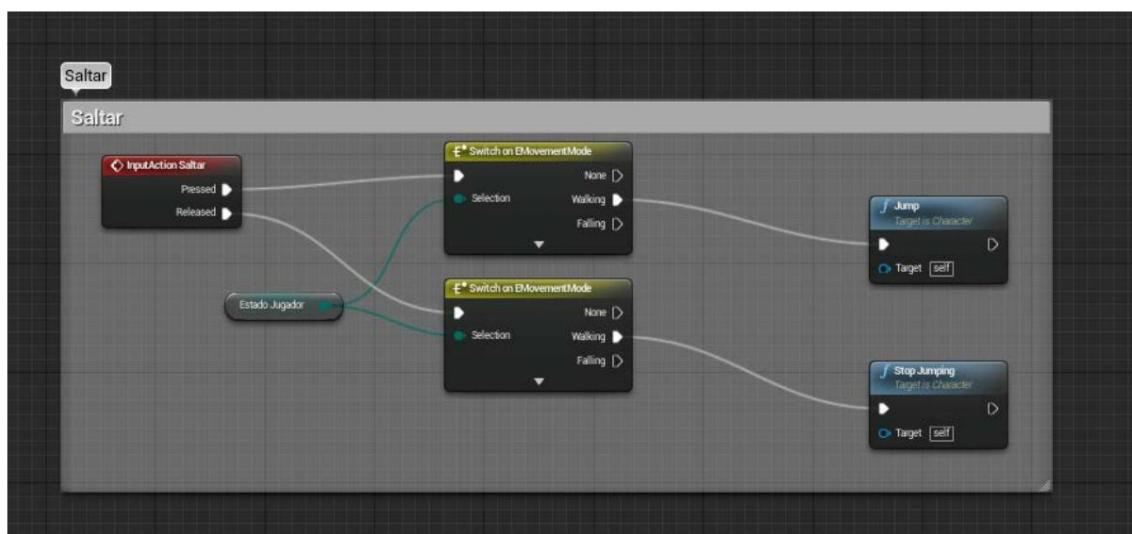


图 103. 跳跃的屏幕截图（仅在步行中）

来源:自制

5.2.1.5.干式制动

干刹车,只能在飞行状态下使用,通过访问
到角色的角色移动组件,其中包含属性和功能
与pawn的运动细节相关,我们将其速度设置为0,0,0。

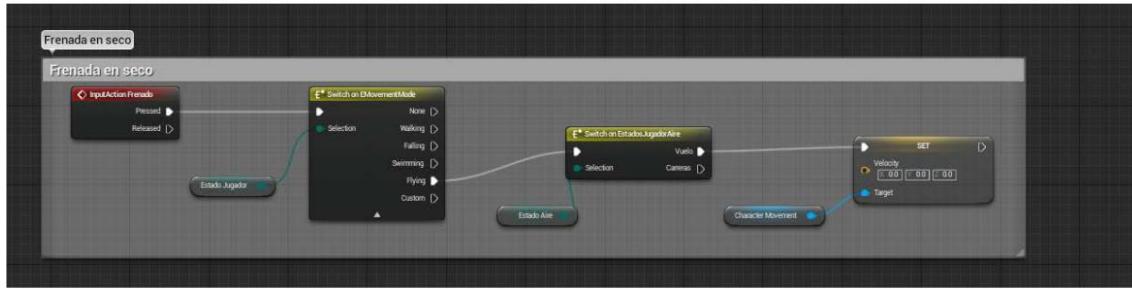


图 104. 使用在飞行状态下执行干式制动的逻辑进行捕获。

来源:自制

5.2.1.6.冲刺纠正

为了让玩家跑起来,我们访问他们的角色移动组件并
我们将他的最大移动速度从 600 更改为 900。

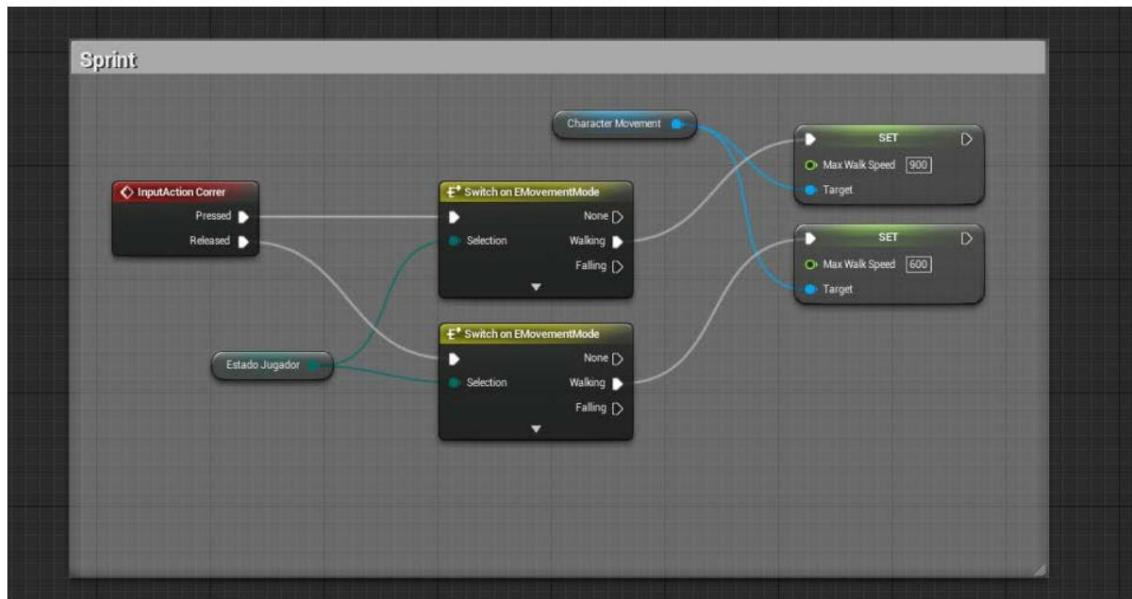


图 105. 带有在 Walking 中运行的逻辑的屏幕截图。

来源:自制

页。 102

5.2.1.7。赛车中的涡轮增压

turbo的逻辑分为两部分,一方面是按下按钮时的逻辑
激活涡轮增压器,它仅在我们处于赛车状态时可用,并且在哪里
我们所做的是当我们按下加速键并且我们的能量不是0时,
我们将最大移动速度更改为赛车状态允许的两倍,并且
我们激活一个允许我们管理能源消耗的布尔值。当玩家
释放涡轮按钮,它会重置飞行的最大移动速度和布尔值。

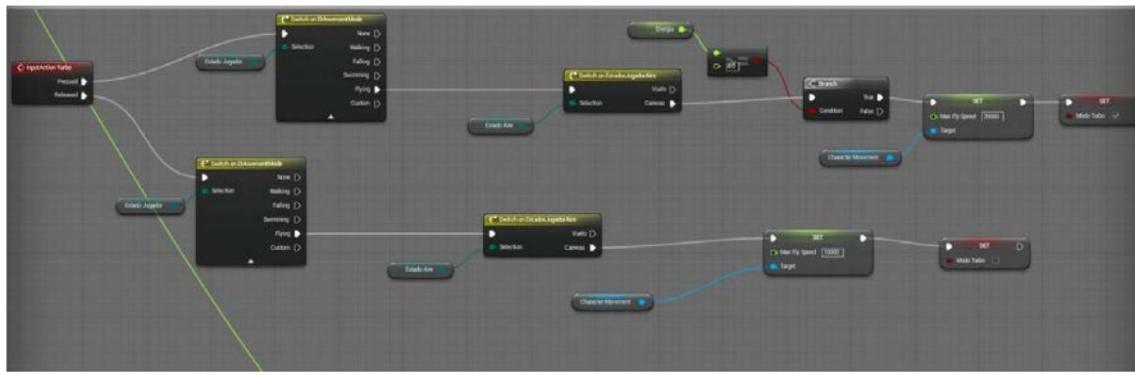


图 106。捕获在 Racing 中执行涡轮增压的逻辑。

来源:自制

在第二部分,涡轮增压,我们每时每刻都在控制能量管理,
我们检查涡轮增压器是否被激活,基于此,我们将消耗能量或
每秒钟一点一点地恢复它。

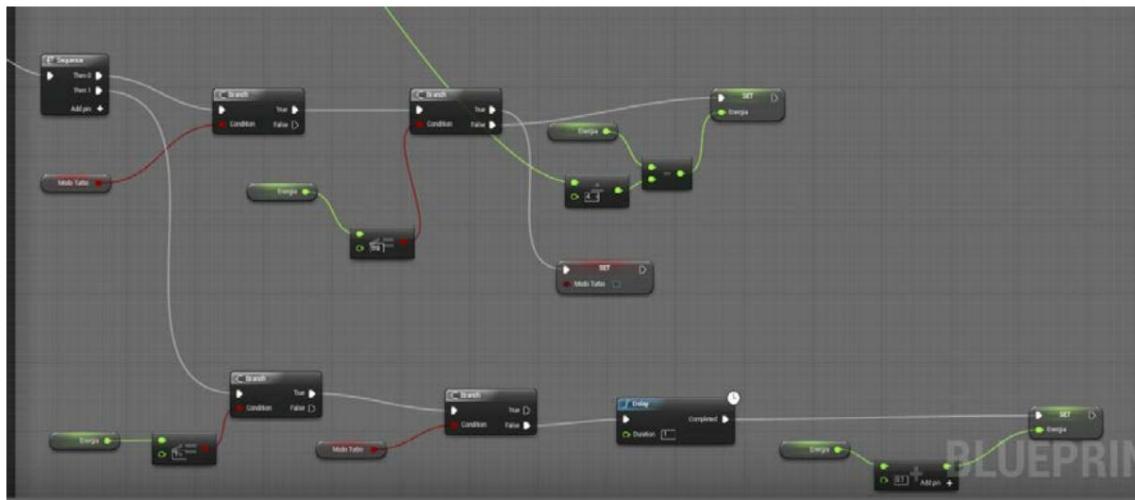


图 107. 减少/增加玩家能量的检查捕获

取决于它是否在使用中。

来源:自制

5.2.1.8.状态变更管理。

玩家可以做出的状态改变可以是手动的也可以是自动的，所以如果你在走路,想飞,你可以通过按键进入那个状态,但是例如,当他在奔跑时突然从孤岛上跌落时,他会自动从estado 走下坡路。

5.2.1.8.1.手动状态更改。

正如我们所说,这些更改是玩家可以通过按下为其启用的按钮或键。

5.2.1.8.1.1.下降手动状态更改

从手动状态到下降状态的过渡可以从飞行状态飞行,也可以从赛车。

为此,我们只需检测玩家实际上处于其中一种状态,并且然后我们改变状态并放置物体受到的重力字符为 1 (它是从 0 到 1 的标准化值)。当我们在一个飞行状态,一旦我们松开潜水按钮,我们就可以返回它。

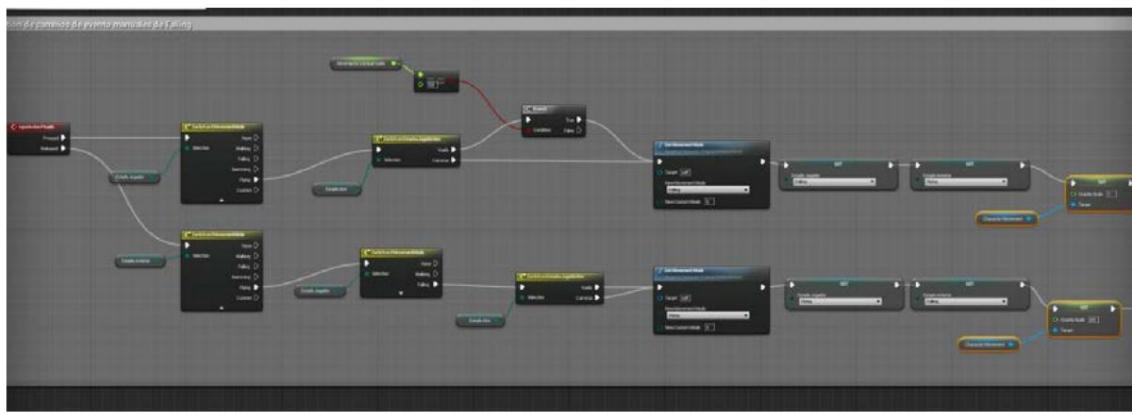


图 108. 捕获 Falling 状态的手动状态变化。

来源:自制。

5.2.1.8.1.2.手动更改飞行状态

要更改航班状态,我们首先要做的,与更改中的相同从以前的状态,是检查我们是否处于可以进行更改的状态,并且如果是这样,我们就这样做,并将玩家所处的重力更改为 0。

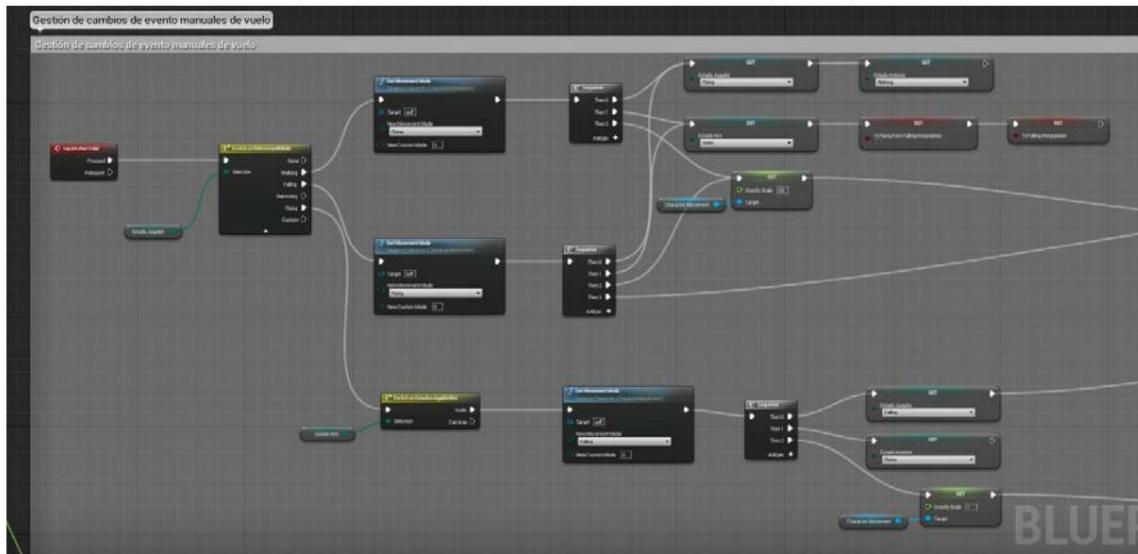


图 109. 捕获手动状态更改为飞行状态。

来源:自制。

当我们再次按下更改为飞行键时,如果我们已经在飞行,我们进入状态
坠落,如果我们在地上,它会自动变成行走,就像已经
我们将在第 5.2.1.8.2 点看到。

5.2.1.8.1.3.赛车手册状态更改

要将状态更改为 Racing,我们首先要做的是,与其余部分一样
状态变化,检查我们所处的状态,如果在这种情况下是飞行,
我们可以改变

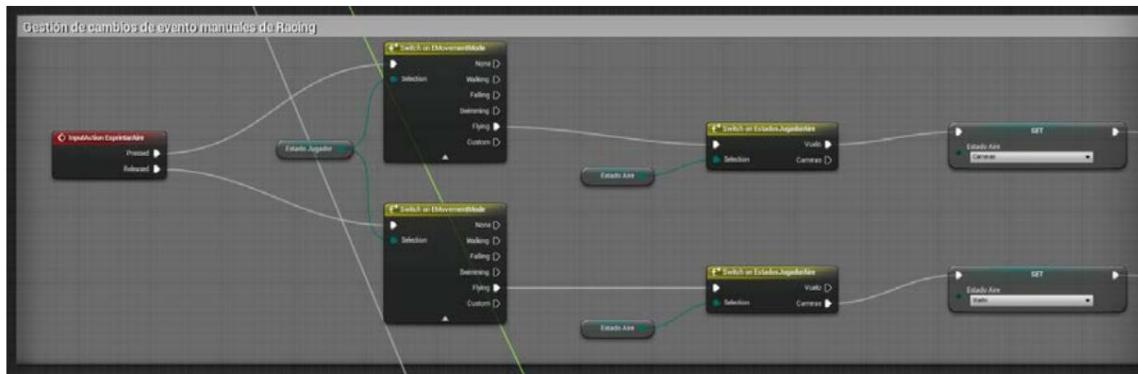


图 110. 手动事件更改检查的屏幕截图以确定它是否可以进入 Racing 状态。

来源:自制。

我们改变这个状态的方法是:

1. 将角色（整个演员）旋转到摄像机面对的位置。
2. 由于我们将旋转整个演员，相机也将随之旋转，所以
我们必须保存它在旋转actor之前的旋转，并应用一次
旋转。
3. 给他一个飞行速度提升，让他在移动时跑得更快。

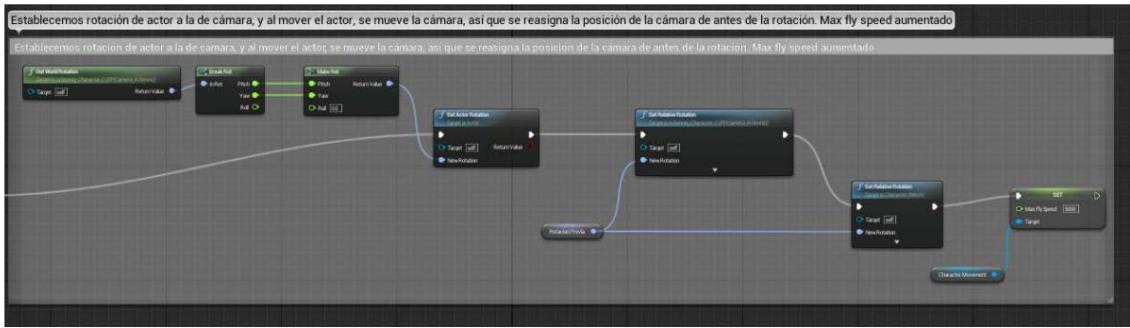


图 111. 传递到 Racing 状态时捕获的逻辑变化。

来源：自制。

4. 在这种状态下不断前进
施加具有一定摩擦力的速度（如果我们不包括摩擦力，当
让我们停止这种状态，它会一直以同样的速度继续飞行
驱动，永远不会停止）。

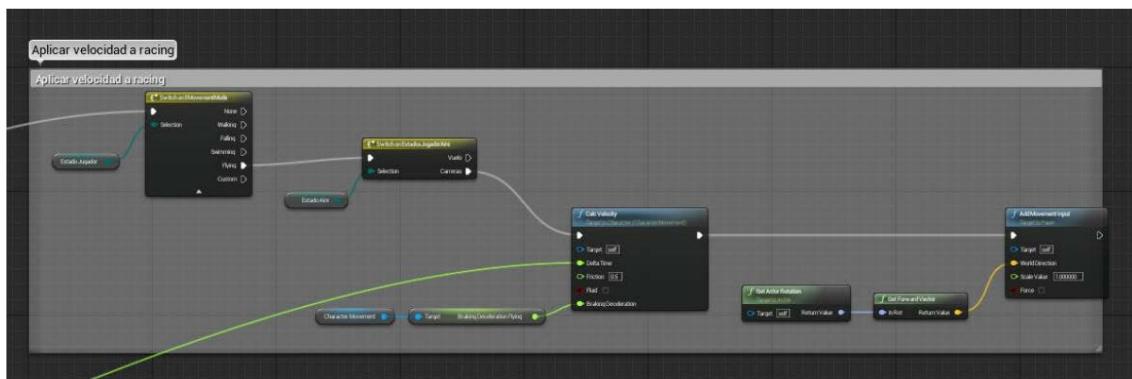


图 112. 进入 Racing 时对玩家应用恒定速度的屏幕截图。

来源：自制。

5.2.1.8.2。管理自动状态更改

至于自动状态变化，正如我们所说，玩家不会
您可以直接使用控制器上的按键进行控制。

5.2.1.8.2.1。行走和跌倒之间的自动状态更改管理

当角色跳下悬崖时,他进入了坠落状态,我们将他旋转 90 度
并且无法进行跳跃或跑步等动作。为了做到这一点,我们检查是否
通过测量其在 z 轴上的速度,角色正在下落,此外,我们检查它是否不是
以相同的方式上升 (我们以这种方式控制它不跳跃),并且还通过
最后,我们检查我们发现自己所处的状态。

如果角色正在行走,我们添加一个延迟来确定根本没有
跳起来,我们再次做同样的检查,如果符合,就会进入下降状态。

如果角色处于坠落状态,我们检查他是否不再坠落 (如果
已经落地),我们让它进入行走状态。

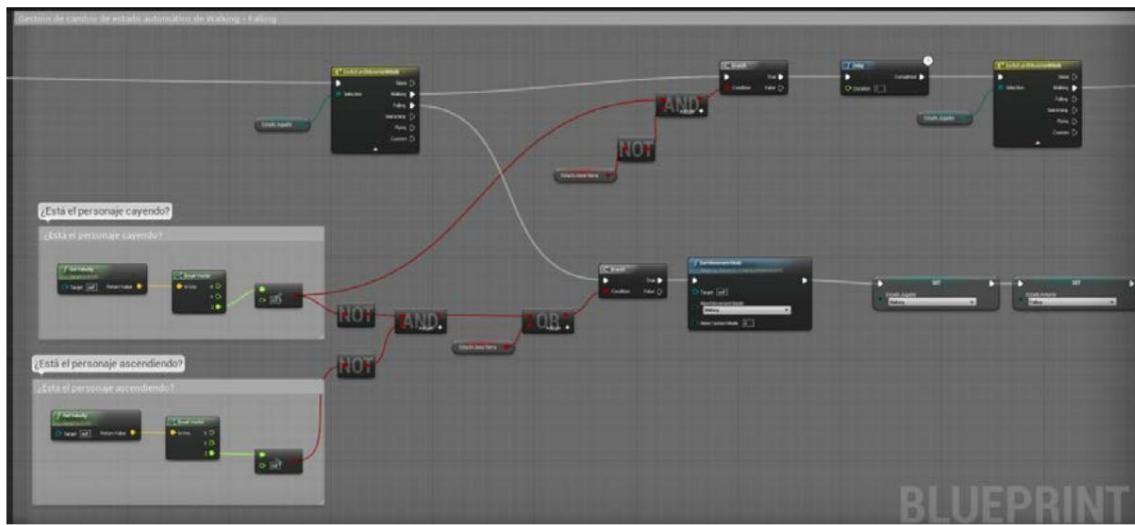


图 113.walking 和 Falling 之间状态自动变化的管理的捕获 (部分

1)。

来源:自制。

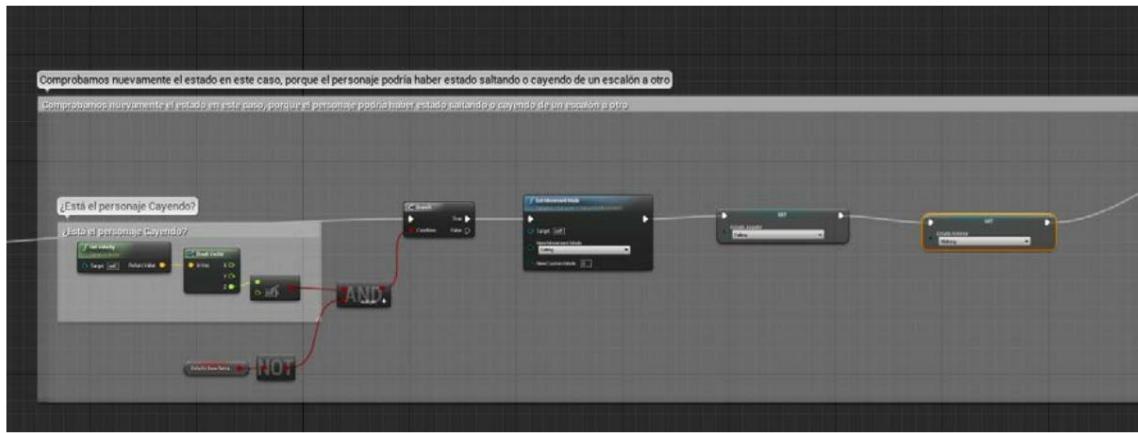


图 114.walking 和 Falling 之间自动状态变化管理的捕获 where

我们再次检查角色是否真的在下降。

来源:自制。

5.2.1.8.2.2。状态变化补间

Tweens 伴随着每个状态变化,让我们在状态变化之间进行更平滑的过渡,例如,如果玩家正在摔倒并撞到地面,他们会遇到旋转 90 度的演员 (胶囊旋转 90 度)和

必须重置此更改 (以便胶囊再次笔直并因此出现起床),通过插值,一点一点地建立我们想要的旋转。

我们的补间主要与事件更改相关,它们是:



图 115. 捕获允许我们传递给不同的布尔变量

角色动作补间。

来源:自制。

5.2.1.8.2.3. 从飞行到下降的自动插值

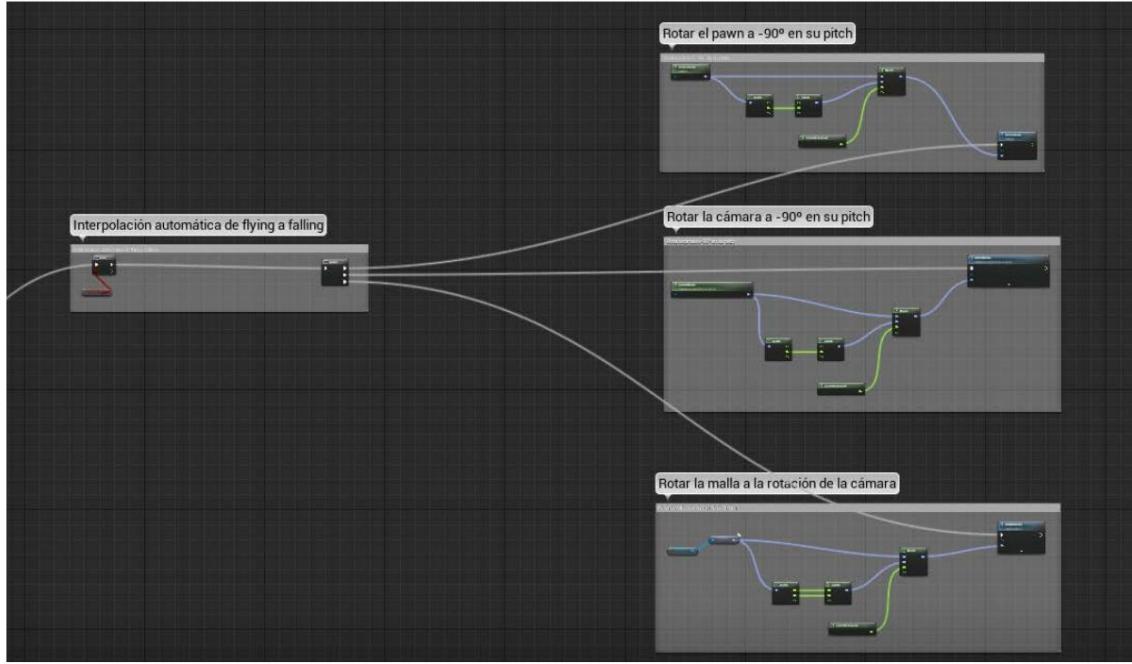


图 116. 捕获从飞行状态到飞行状态的插值的一般逻辑
下降。

来源：自制。

从飞行到下降的插值基本上包括以下步骤：

1. 检查必须做的事情：

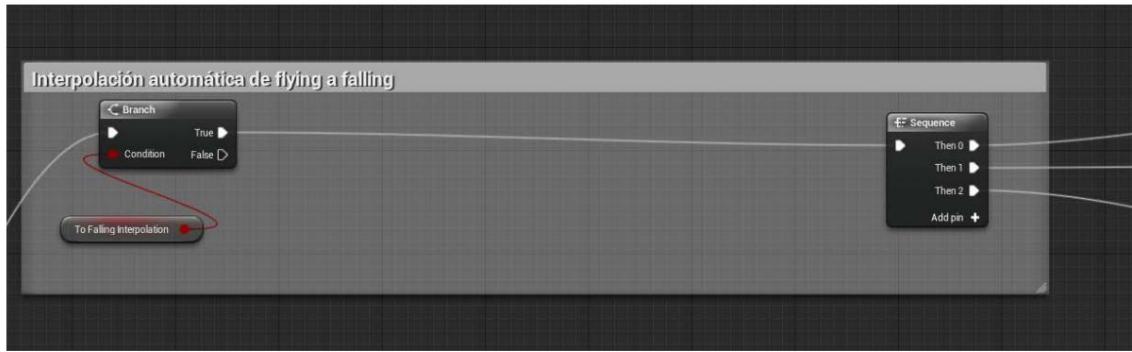


图 117. 确定是否应用
从飞行到下降的插值。

来源：自制。

2. 将字符旋转 -90° 在其音高添加插值。

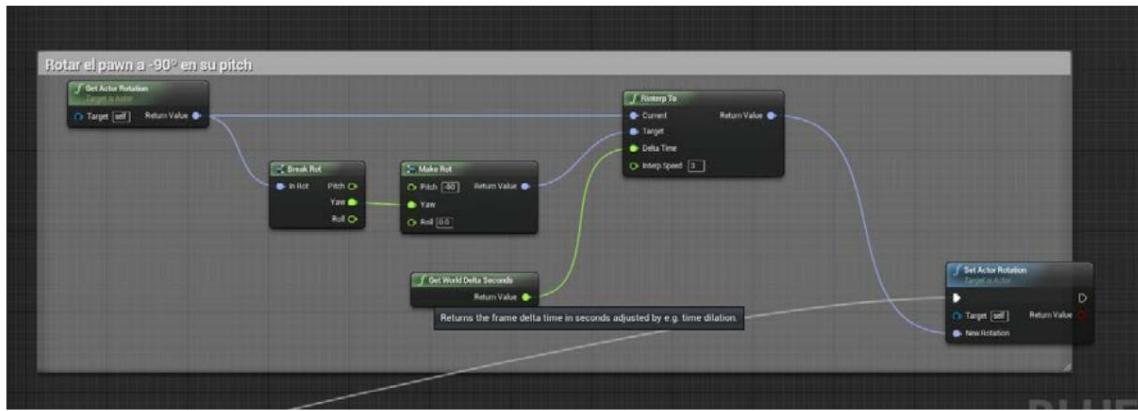


图 118. Pawn 旋转步骤捕获以模拟跌倒。

来源:自制。

3. 将摄像机旋转 -90° 并添加插值 (如果演员旋转 -90°,

我们希望相机旋转 -90° 它的视点,即使我们转动 pawn,它也会
跟随相机对焦同一个地方)

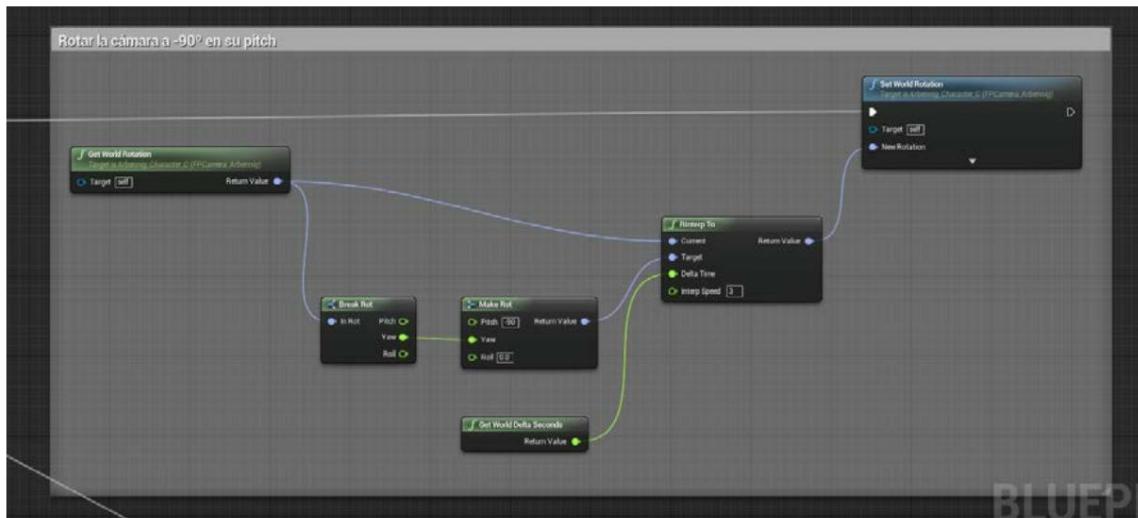


图 119. 捕捉摄像机旋转以模拟 pawn 正朝下落

下。

来源:自制。

4. 也使用插值将网格旋转到相机的旋转。

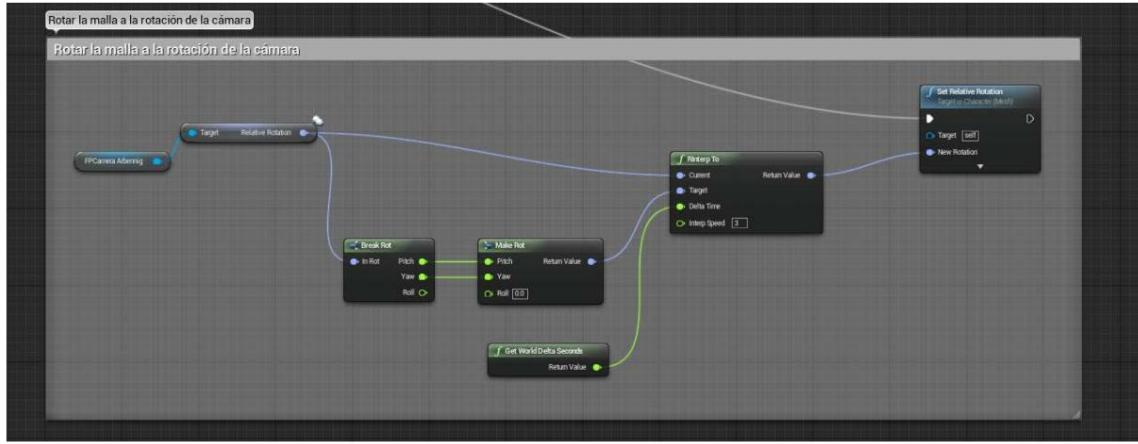


图 120. 代码的旋转网格到网格旋转部分的旋转捕获。

来源:自制。

5.2.1.8.2.4。从下降到飞行的自动插值

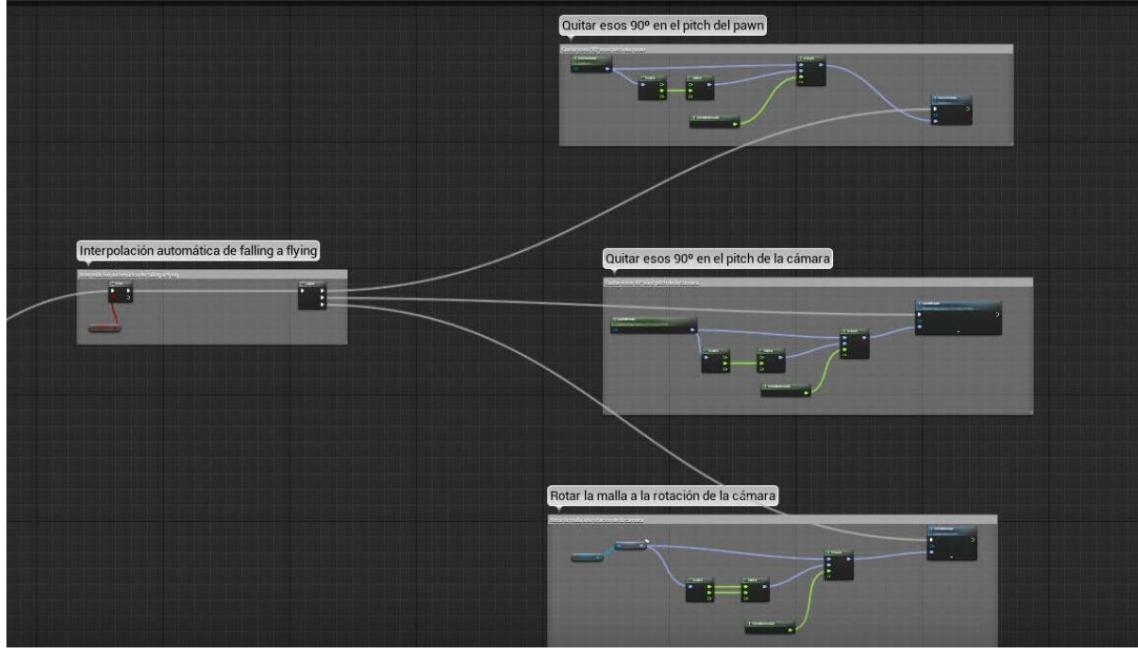


图 121. 捕获从下降到飞行的插值的一般逻辑

来源:自制。

对于从下降到飞行的插值,步骤与从飞行到完全一样

下降,但不同的是这次我们删除了我们添加的 -90°,留下了它们
在 0°。



图 122. 检查捕获是否需要从飞行到下降的插值

来源:自制。

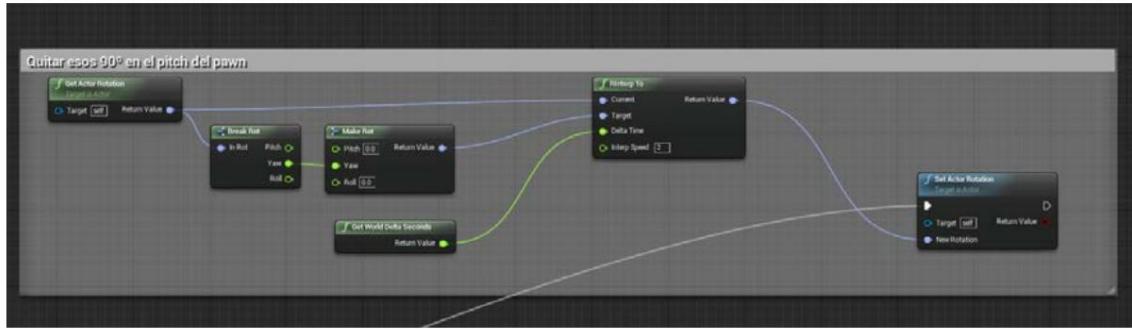


图 123.在角色坠落状态之前恢复旋转的屏幕截图。

来源:自制。

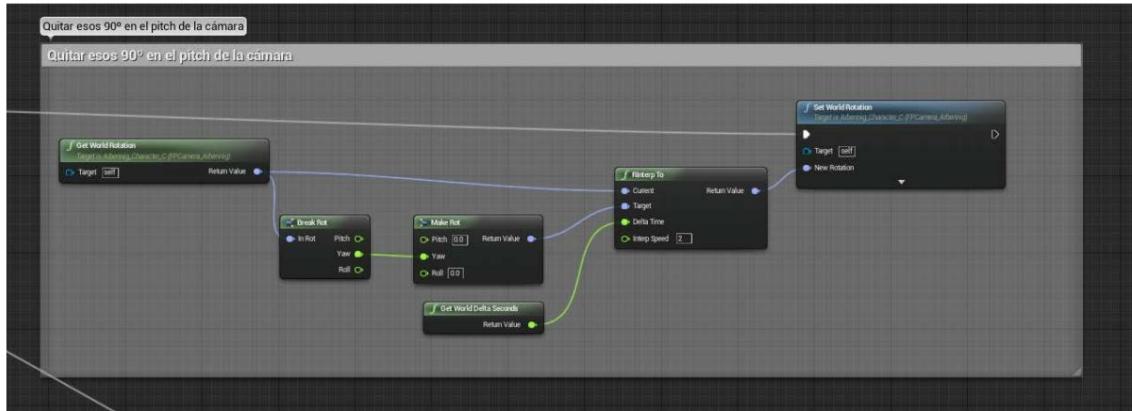


图 124.在相机下降状态之前恢复旋转的捕获。

来源:自制。

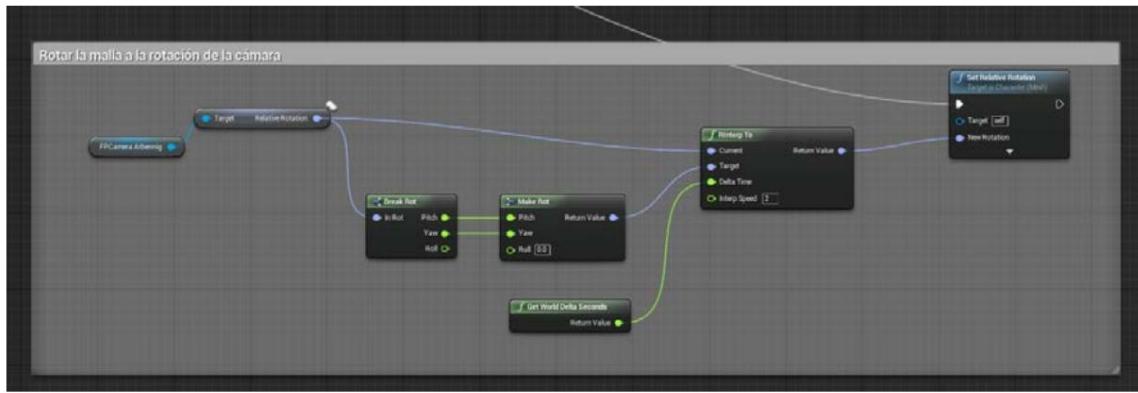


图 125. 网格旋转捕获到相机旋转

来源:自制。

5.2.1.8.2.5。从赛车到飞行的自动插值

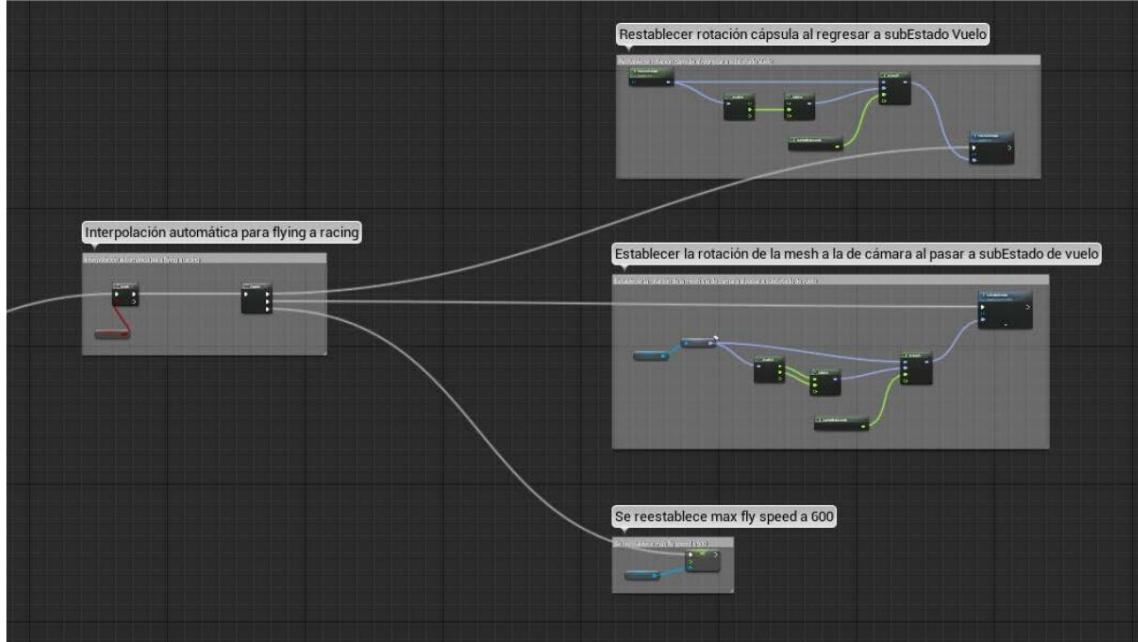


图 126.捕获从飞行到赛车的自动插值的一般逻辑。

来源:自制。

对于从 Flying 到 Racing 的插值,我们执行的过程是重置旋转

胶囊 (角色)的音高和滚动为 0,因为在赛车中这些值会发生变化。

之后,我们将网格的旋转分配给相机的旋转 (当我们

在赛车中,网格随着演员旋转,而相机是空闲的),最后,我们降低
最大飞行速度为600。

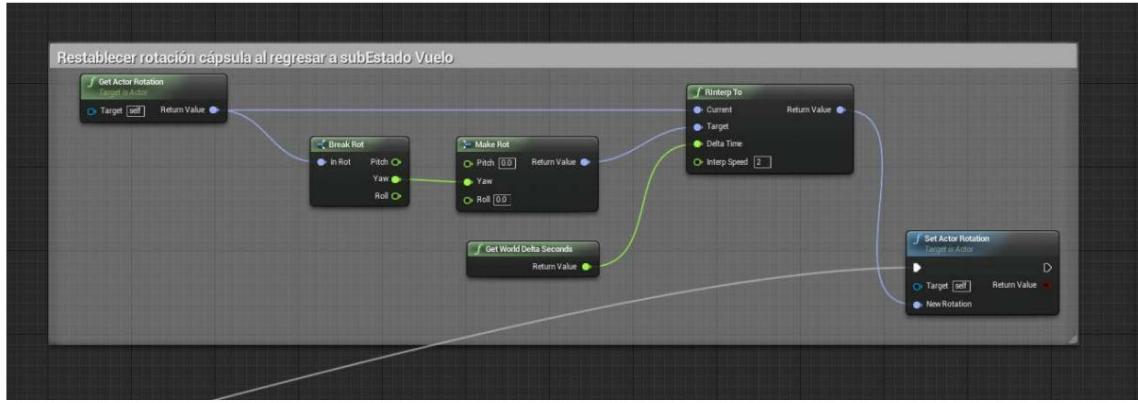


图 127. 捕获以在返回飞行时重置胶囊的旋转。

来源:自制。

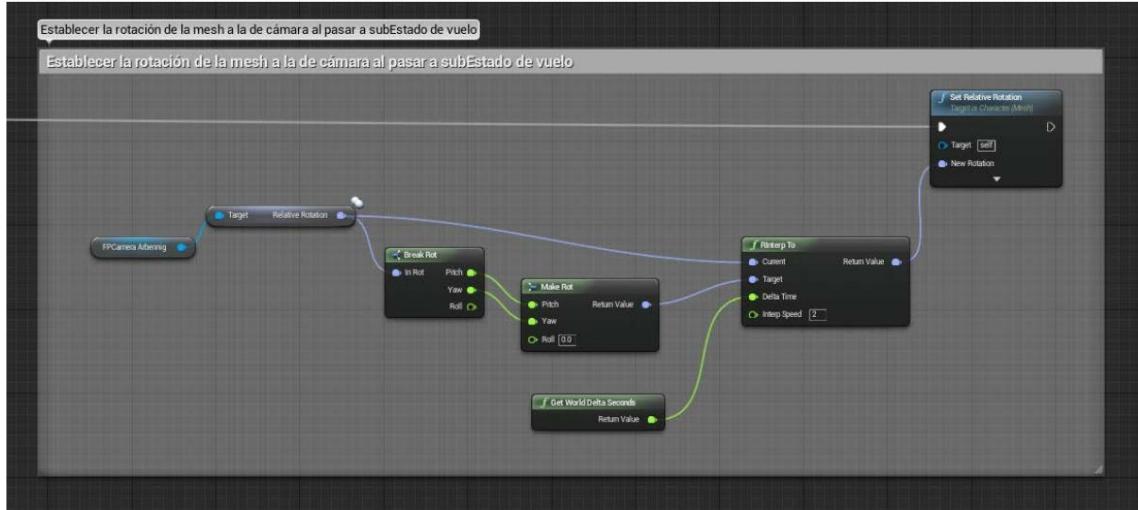


图 128. 悬停时设置网格旋转到相机的逻辑截图

(没有比赛)。

来源:自制。

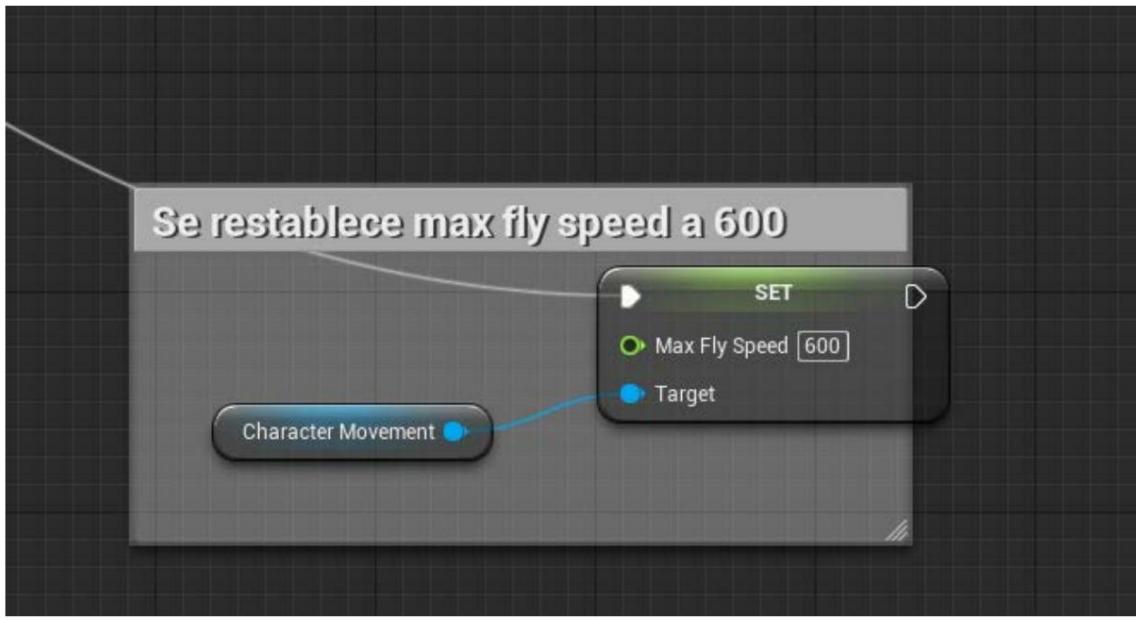


图 129. 将最大移动速度重置为 600 而不是 10000 的屏幕截图

就像我们在赛车中一样。

来源:自制。

5.2.1.8.2.6.激活补间

作为在插值方面添加的最后一个细节,我们激活它们以执行

之前命名的自动和手动状态更改的更改,我们通过将这些变量之一设置为 true 并将其余变量设置为 false 来做到这一点。

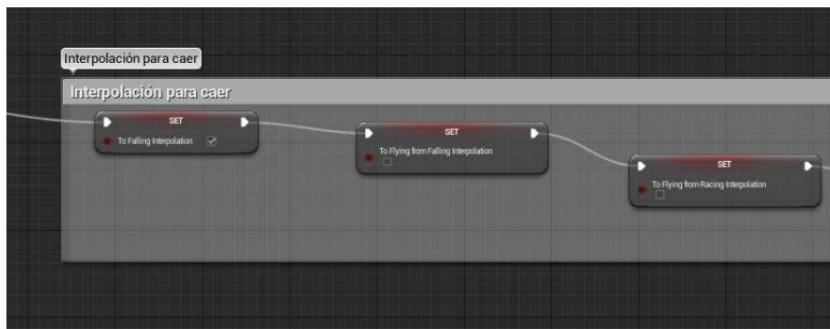


图 130. 跌倒插值激活捕获（从步行到跌倒）

来源:自制。

5.2.1.9。与 Oculus Rift DK1 的兼容性控制。

为了完成与 Oculus Rift 相机兼容的游戏控制器的制作

(请记住,对于 Racing 状态,我们已经在上一节中进行了一些更改,并且
我们已经激活了允许我们使用它的插件),在我们的特殊情况下,我们决定
进入玩家控制器,然后从那里管理我们角色的状态
行走和飞行,使网格根据外围的相机移动,这
将分配给 PlayerCameraManager。

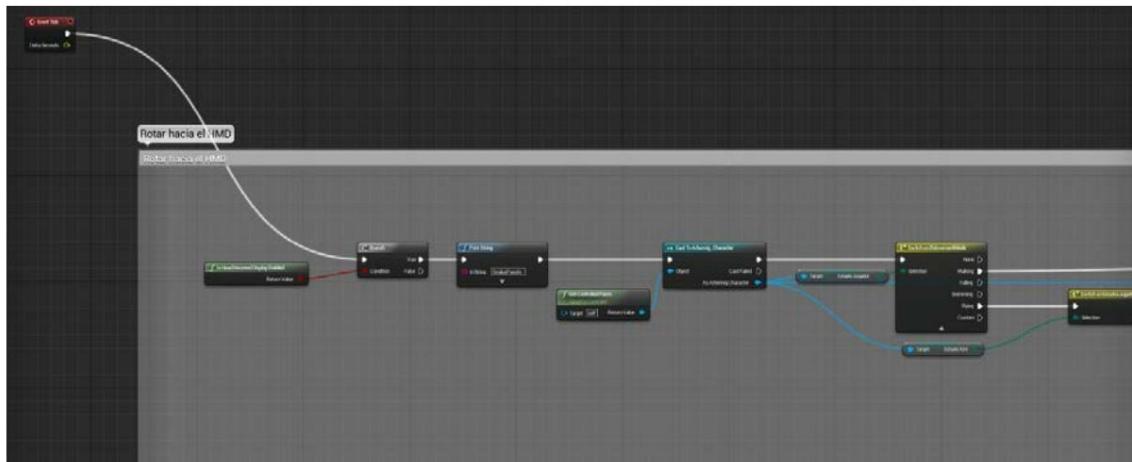


图 131. 在 PlayerController 中捕获必要的检查,以便相机
旋转到 Oculus Rift 设备旋转的位置。

来源:自制。

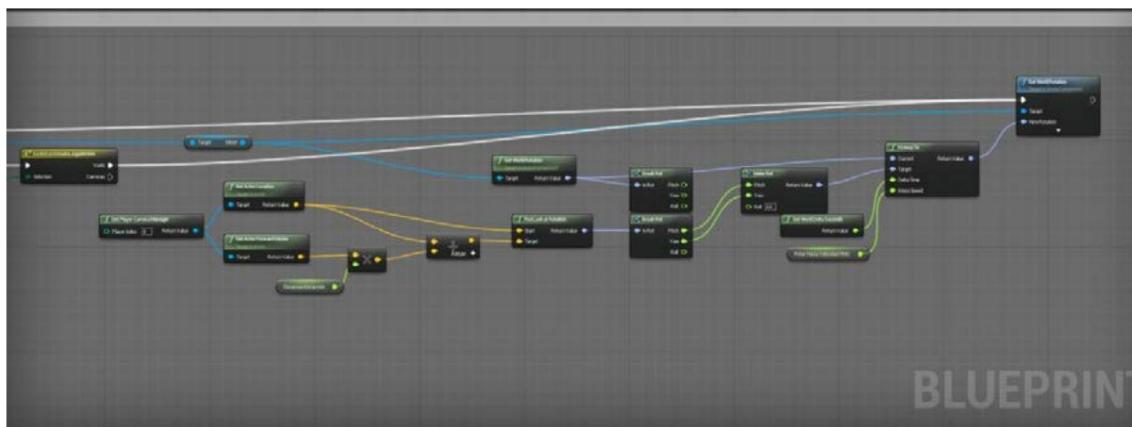


图 132. 捕获逻辑以使网格沿 Oculus 指示的方向旋转
步行和飞行。

来源:自制。

5.2.3。游戏级开发。

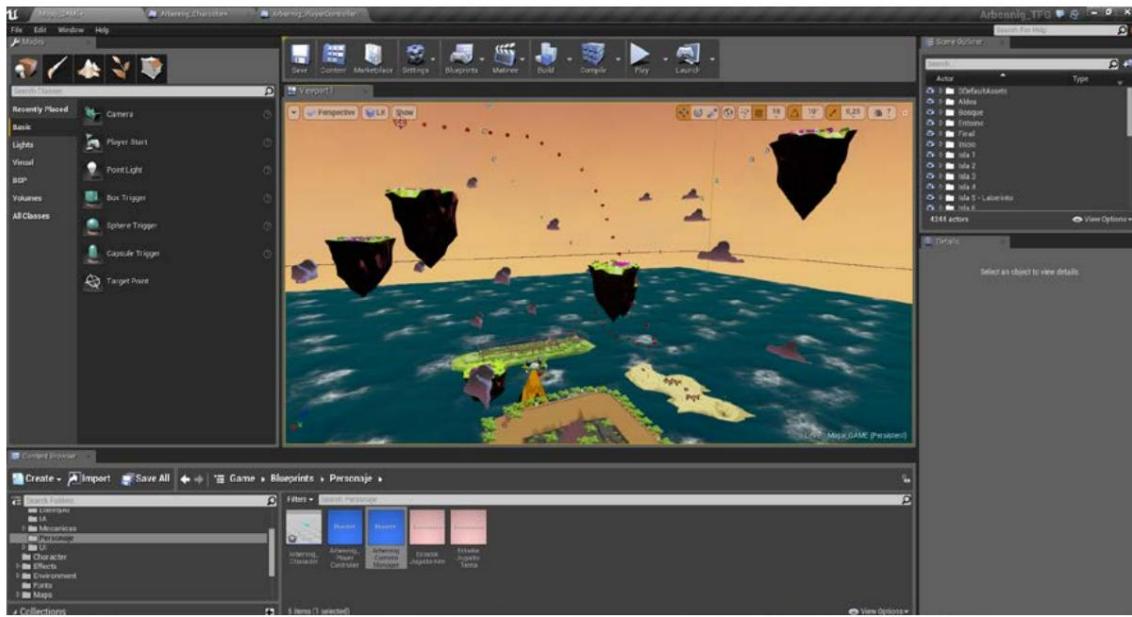


图 133. 打开 Arbannig 项目的 Unreal Editor 的屏幕截图。

来源:自制。

本节旨在解释游戏中所有未实现的组件的实现
它们带有干扰游戏流程的实现逻辑。也就是说,导入和
资产定位、创建地形和粒子效果以及照明。

5.2.3.1。资产创建和导入管道。

在这个项目中,我们包括从其他模板和材料带来的资产
虚幻,作为我们在 3D 建模程序中创建的我们自己的资产
之前。

下面解释如何执行这两项任务。

5.2.3.1.1。从 3ds max 导入资源

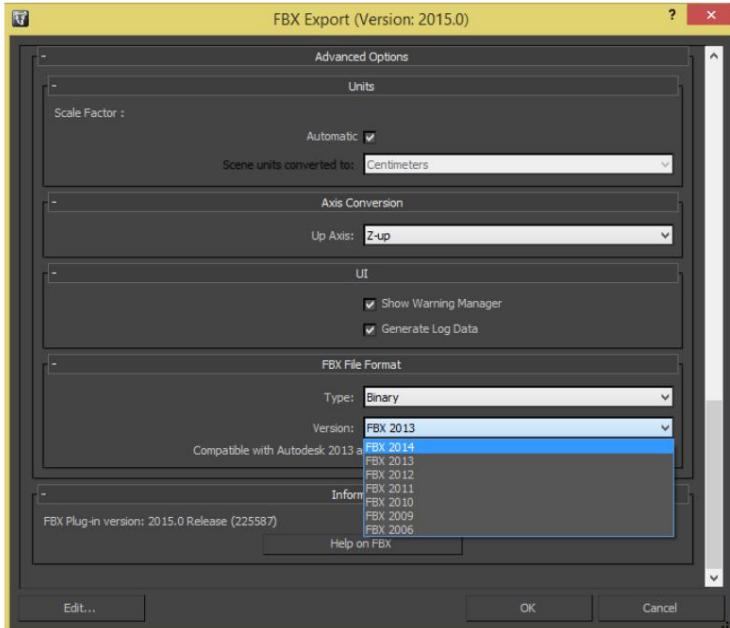


图 134. 选择 FBX 2014 从 3ds max 导出屏幕截图。

来源:自制。

要将网格导入引擎,我们必须做的是将它们导出为 .fbx 格式的版本 2014[26],并在导出文件后,将其拖到引擎的内容浏览器中,或者如果喜欢,单击导入按钮并选择文件。

引擎会识别它,并向我们显示一条导入消息,我们只需要在其中单击导入,因为默认情况下,我们最初对拥有所有导入的网格。

如果稍后,例如,我们对特定碰撞中的网格不感兴趣,我们可以访问到已经导入的静态网格并选择删除碰撞。

当我们导入一个 fbx 时,它已经分配的材料会自动分配在 3ds max 中,在引擎中自动创建。

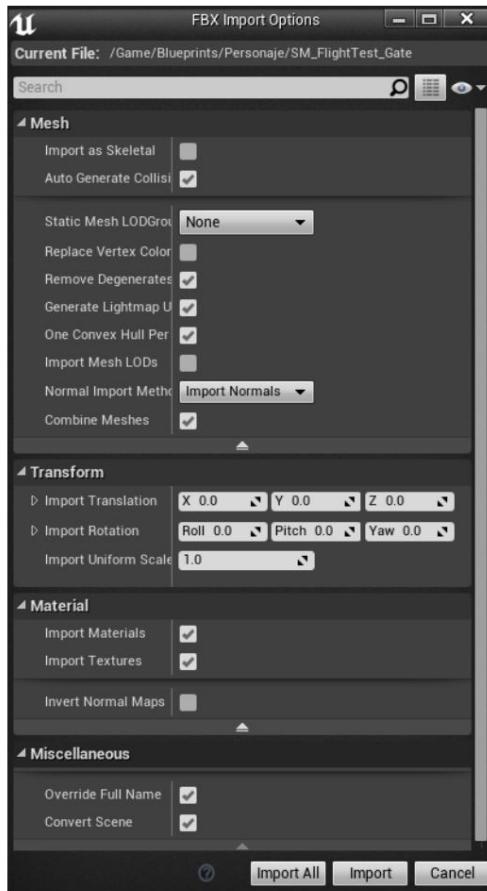


图 135. 尝试导入 .fbx 网格后出现的消息屏幕截图

来源：自制。

导入网格后，我们可以自由地将它们放置在我们的舞台周围
从代码中拖动或生成它们。

5.2.3.1.2。从另一个 UE4 项目导入资产

如果我们想从其他虚幻引擎 4 项目中导入资源，例如从模板
提供学习和示例引擎功能，这些都是免费的，
我们可以通过单击资产或资产集并在其选项之间进行选择来完成
Migrate 选项，它将要求我们提供迁移它们的目录。

为此，我们需要从相同版本的引擎迁移资产，因为
来自更高版本的资产可能无法工作并导致问题。

5.2.3.2。创建项目的土地（景观）。

可以使用编辑器中包含的景观工具来使用引擎创建地形。

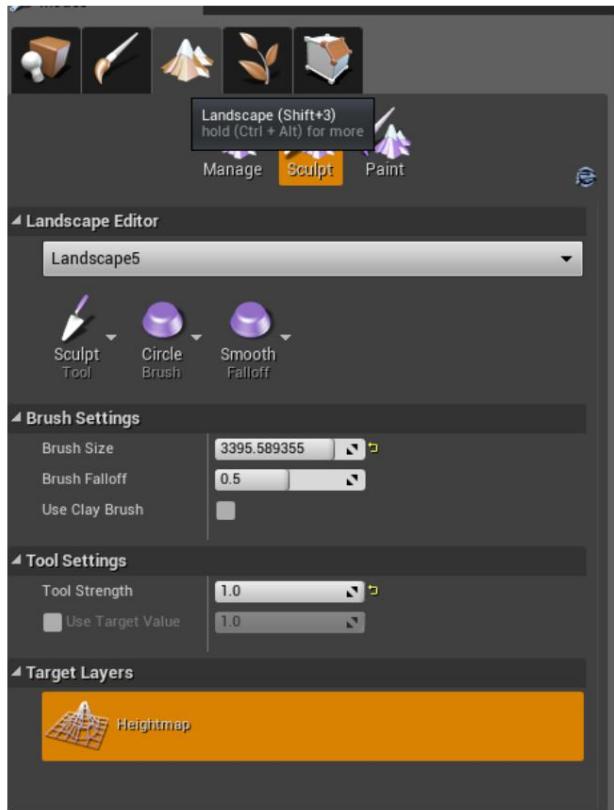


图 136.捕获引擎的横向工具。

来源:自制。

它的操作很简单,使用管理选项,我们可以创建新的景观,
我们在其中看到创建和选择要在其中使用的材料的不同选项。

使用雕刻工具,我们可以“建模”这个最初是平面的景观,
我们可以用它提供的画笔自由地做,或者加载一个高度图和
自动制作所需形状。

最后,借助 Paint,我们可以用材质绘制风景。

在这个项目中,我们使用景观工具来创建不同的岛屿,而不是
构成德尔弗里多兰世界的花车。前往迷宫所在的岛屿
Totem 5,已经创建了一个基本的高度图,并在引擎中创建后对其进行修饰。

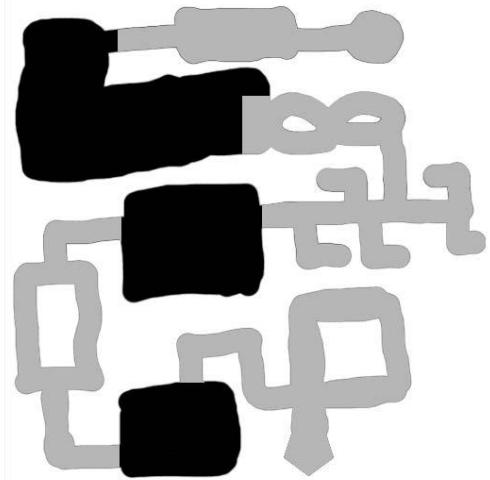


图 137. 5 号岛迷宫的高度图。

来源：自制。

在这个应该是灰度的高度图中，黑色表示最大深度，
而白色意味着不需要深度。



图 138. 使用高度图工具创建迷宫后的景观
景观的。

来源：自制。

其余的岛屿是自由创建的，直到它们最终形成合适的形状。

5.2.3.3。创建火火花的粒子效果。

我们想使用火花粒子系统作为装饰元素
等级。这些闪光给火山形装饰带来更真实的感觉,比
通常出现在每个图腾旁边。

5.2.3.3.1。粒子系统的解剖。

在开始创建粒子系统之前,我们有必要了解不同的部分
虚幻引擎 4 中的粒子系统。

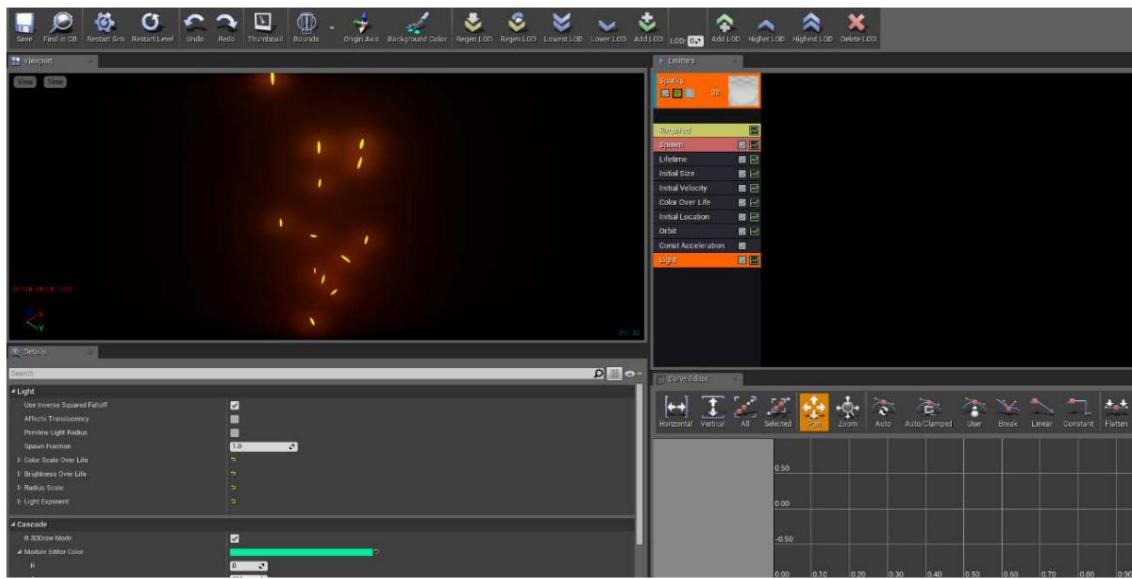


图 139. 捕获已创建的粒子系统。

来源:自制。

也就是说,这个引擎中的粒子系统是由发射器和一个
发射器由一个发射块和一个模块列表组成。

发射块包含帮助我们创建不同组件的不同属性
的粒子系统。这些属性是块的可见性（显示或不显示
块）,在预览窗口中看到的发射形式（可以是普通形式,只有
点,只有刻度,因为它会在舞台上看到（点亮）,不显示任何东西）,以及“独奏”按钮
(当我们有多个发射器时,它允许我们只可视化我们用 Solo 标记的那个)。

关于模块列表,我们有Required (它将允许我们更改属性,例如
发射材料,您还可以更改发射的方向和旋转),
Spawn (允许我们配置要发射的粒子数量 (在下拉菜单中)
Rate->Distribution->Constant)), Lifetime, Initial Size, Initial Velocity, y Color Over Life, como
默认为模块,但可以通过右键单击添加或删除模块
模块列表 (除了必需的所有模块都可以从发射器中删除)。这

模块允许我们“暴露”它们中的每一个,这允许我们改变它们的属性
随着时间的推移在编辑曲线中。每个模块的工作流程是
首先访问属性,然后,如果需要触摸编辑曲线,请通过
那时给她。

一个粒子系统可以由多个发射器组成。创造更多的发射器
粒子,它可以通过右键单击发射器窗口来完成,它允许我们选择
(创建精灵发射器),要删除它,右键单击发射器,然后在选项卡中
发射器,允许我们不同的选项,其中我们找到删除发射器。每个发射器被分成列,称为发射器列。

现在已经解释了引擎中粒子系统的操作,我们继续创建我们将在游戏中使用的粒子系统,我们将创建一些装饰性火花,它们属于精灵发射器类型。

5.2.3.3.2. 实施火花系统。

在开始创建粒子系统之前,请记住它实际上是
它由两部分组成,一方面我们有视觉(我们如何看待效果本身),另一方面,
它的行为(它背后的逻辑,由上面描述的
解剖学)。

创建粒子系统的一步是在内容浏览器中创建它,
这可以在您创建蓝图时完成(右键单击->粒子系统),然后给它一个
姓名。

接下来就是设置它的基本参数,也就是通过Required模块。

记住,火花需要发射基体材料。我们创造了这个基础材料
我们,在我们的特殊情况下,它将是一个简单的材料:

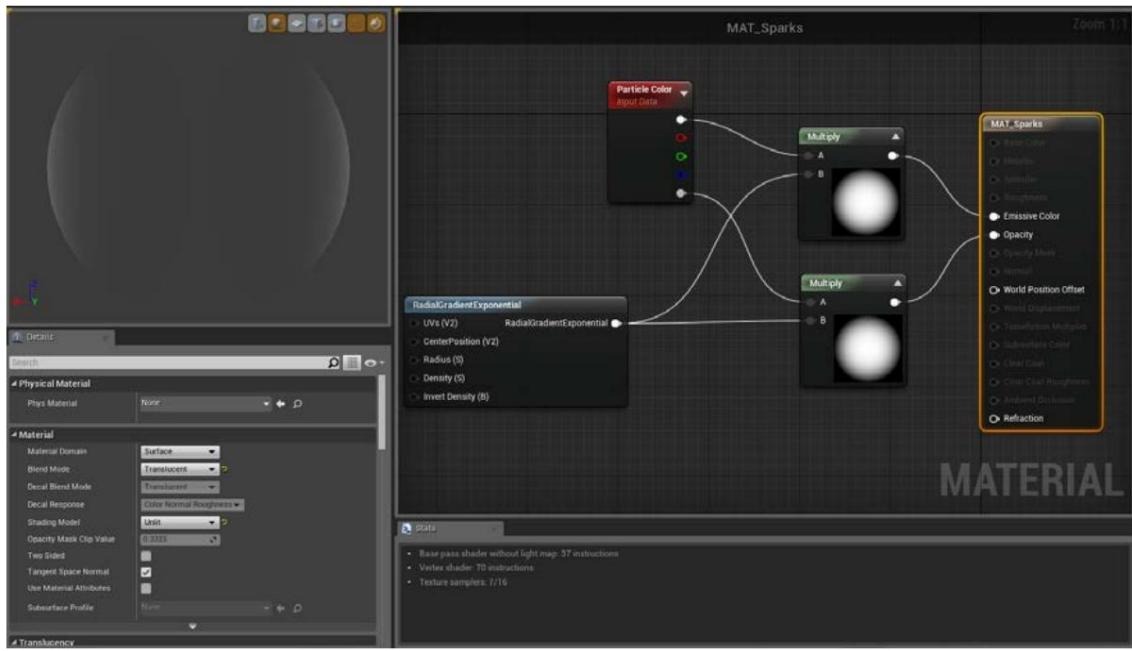


图 140. 使用创建的粒子系统捕获材质。

来源：自制。

与普通材质相比,这种材质的特殊之处在于它使用 ParticleColor 类型的 InputData 事件接收数据。另一方面,为了能够正确地创建材质,我们需要在材质的属性中设置它的混合模式是半透明的（这就是

启用我们的 opacity 属性）,以及我们将其设置为 unlit 的着色模式,因此我们的粒子系统对 CPU 尽可能没有攻击性。

一旦我们有了材料,回到我们的火花粒子系统,我们改变它在必需模块中。

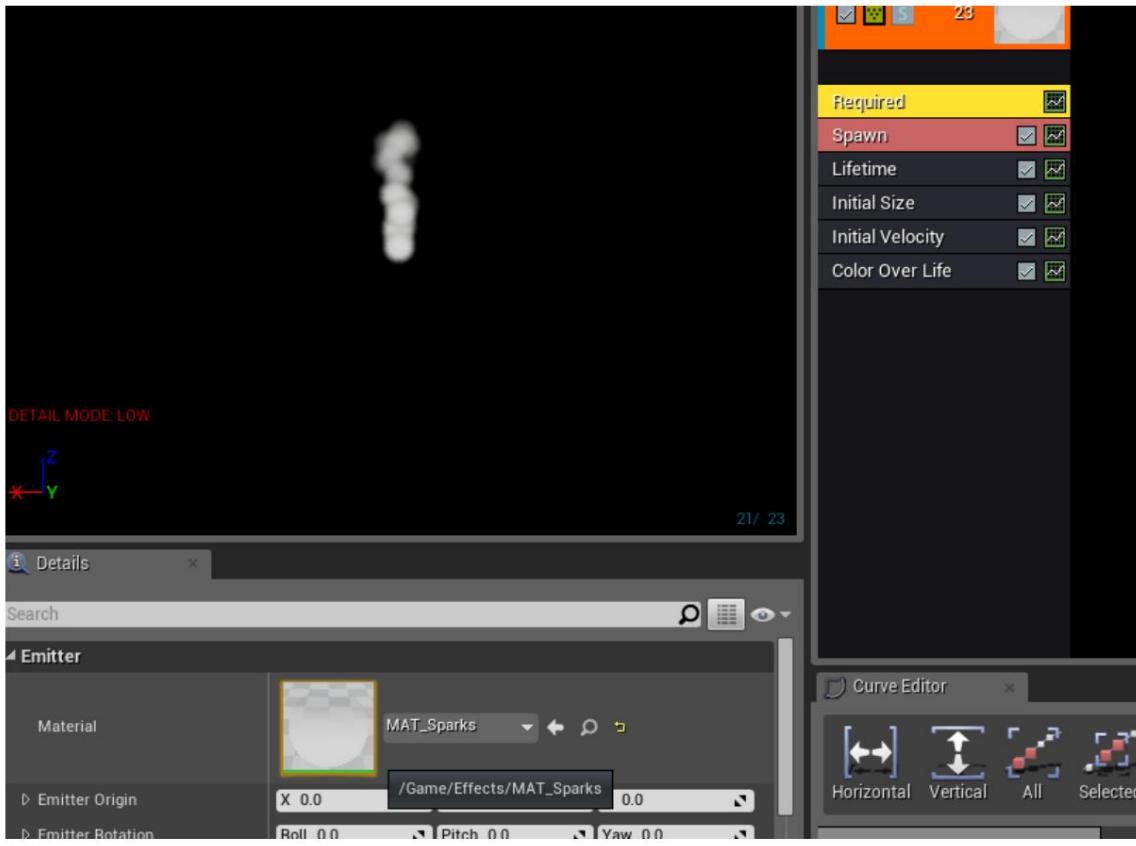


图 141. 应用创建的材质后捕获粒子系统。

来源:自制。

在我们的例子中,我们希望发射面朝上,所以我们不打算修改发射器的旋转,我们也不想改变它的原点。

我们将 Screen Alignment 设置为 PSA Velocity。屏幕对齐用于更改每个发射粒子的容量。默认情况下,它在 PSA Square 上,发行单每个发射粒子的正方形多边形,我们想要粒子的 PSA Velocity 改变,并在必要时迅速改变。



图 142. 屏幕对齐选项屏幕截图

来源:自制。

我们进入 Spawn 模块,因为我们不需要默认粒子 (20),但是少就够了,我们把它改成10。

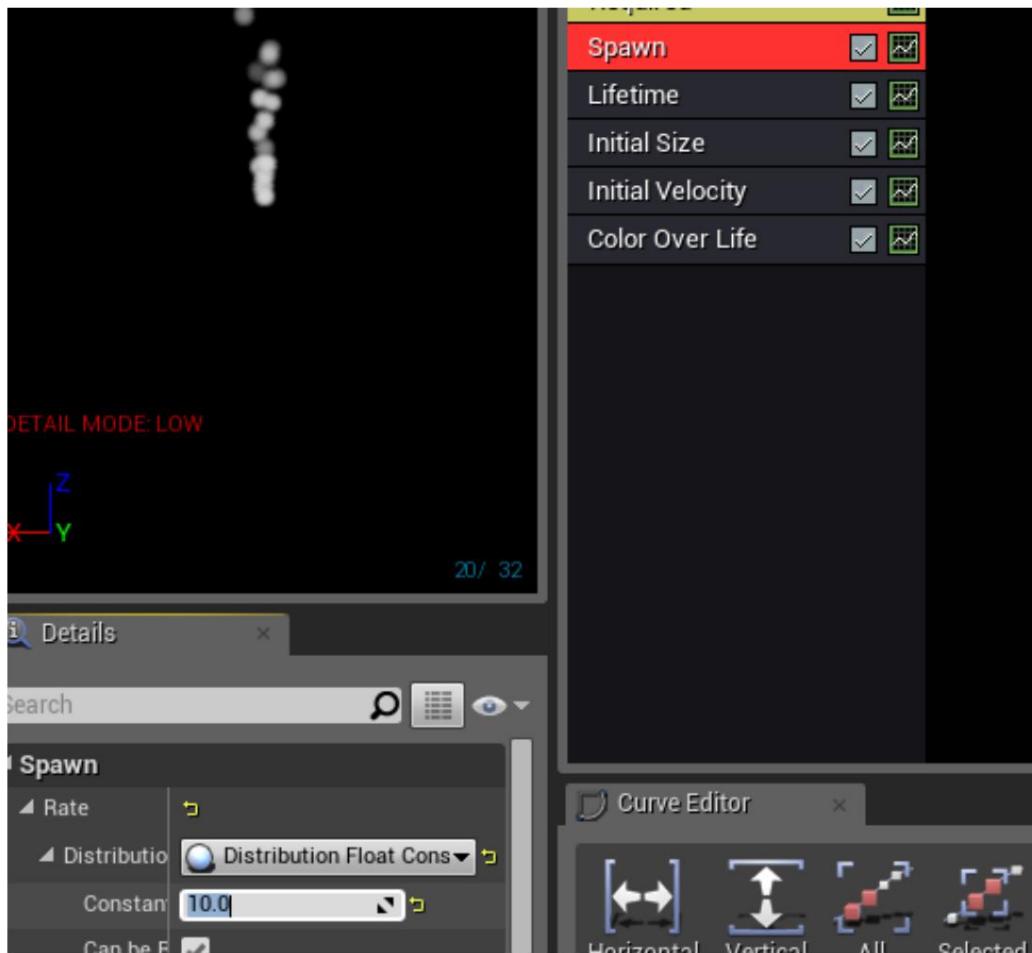


图 143.默认情况下将粒子系统的数量减少一半后捕获的粒子系统。

来源:自制。

我们继续到 Lifetime 模块,在这个模块中我们有最小和最大持续时间,默认情况下两者都出现在 1。我们将最大时间更改为 3,这样一些最后时间多,别人少,而且不那么统一。

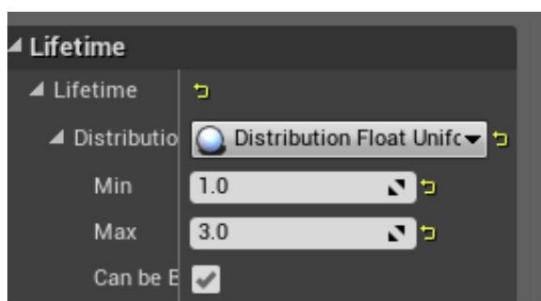


图 144. 捕获我们决定分配的最小和最大生命周期。

来源:自制。

在用于改变粒子大小的 Initial Size 模块中,我们将给它们最小和最大尺寸相等,以及值 (2,10,2) ,这使它看起来更像火花。

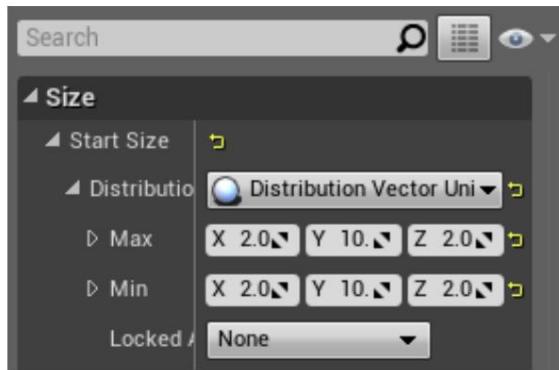


图 145. 捕获修改为最适合的粒子的初始大小
通缉。

来源:自制。

现在我们继续配置 Color Over Life 模块,该模块用于修改粒子的整个生命周期。为此,我们有一个带有两点的向量 0
1.颜色的起始值为0,结束值为1。



图 146. 捕获粒子应具有的所选颜色。

来源：自制。

我们将以下 (50,5,0) 作为输出值 (out val) 放在 0 中，并在第 1 点放置 (50,10,0)。当我们在出现在我们面前的 RGB 向量的每个位置从 1 开始时，它开始创建发光效果。



图 147. 重载值后捕获的结果，可以看到 Glow 效果。

来源:自制。

下一步是在发射块中再添加一个模块,即 Initial Value 模块(什么我们在模块列表中单击右键添加)。有了这个模块,我们将制作原点的火花出现“熄灭”,因此粒子开始看起来更加分散。

在我们的例子中,我们通过将最大分布更改为(30,30,0)并将最小值更改为(-30,-30,0)来做到这一点。

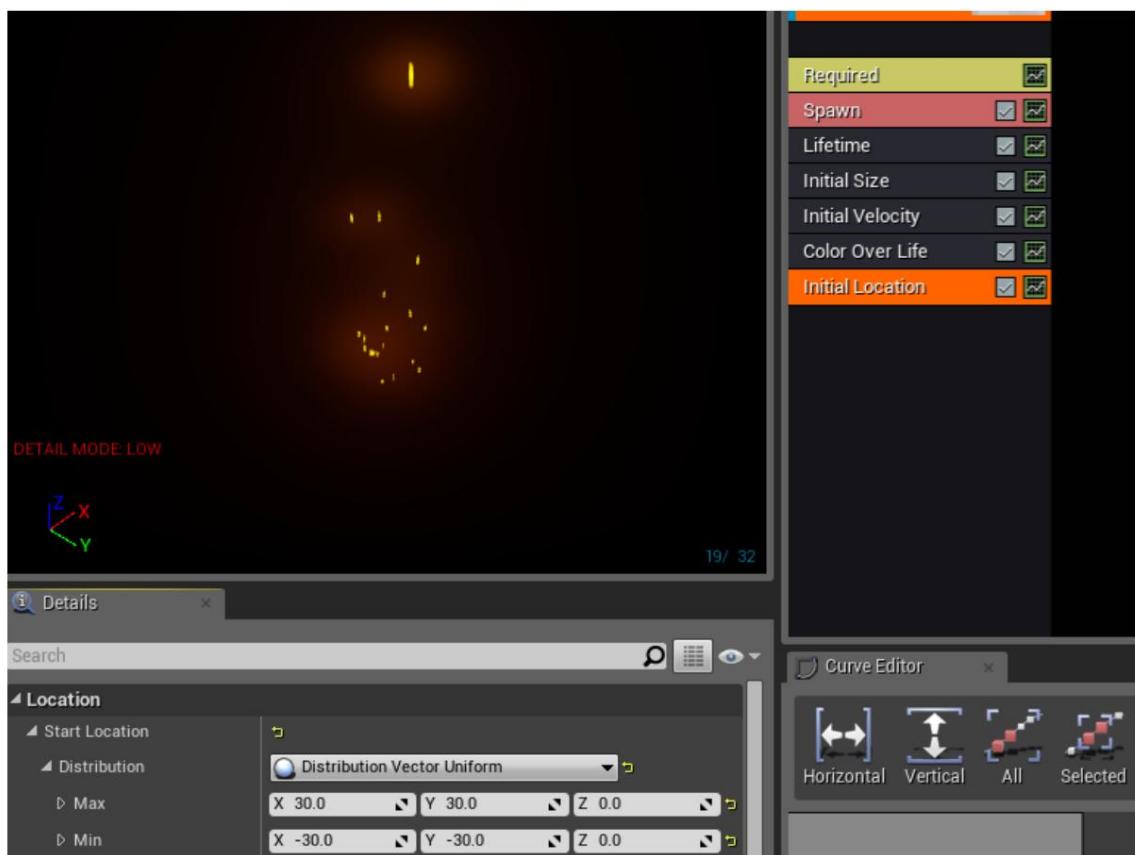


图 148. 修改初始位置后捕获的粒子系统。

来源:自制。

在此之后,目标是让它们看起来更像火花,稍微多一点混乱的。为此,我们将添加轨道模块。此外,我们希望它给人的感觉是它们通过提高速度来提高,我们通过添加模块来做到这一点
const 加速度,并在 Z 中的加速度中指示值为 100,因此,在其时间生命中,粒子会不断增加它们在 Z 方向的速度,直到它们消失。

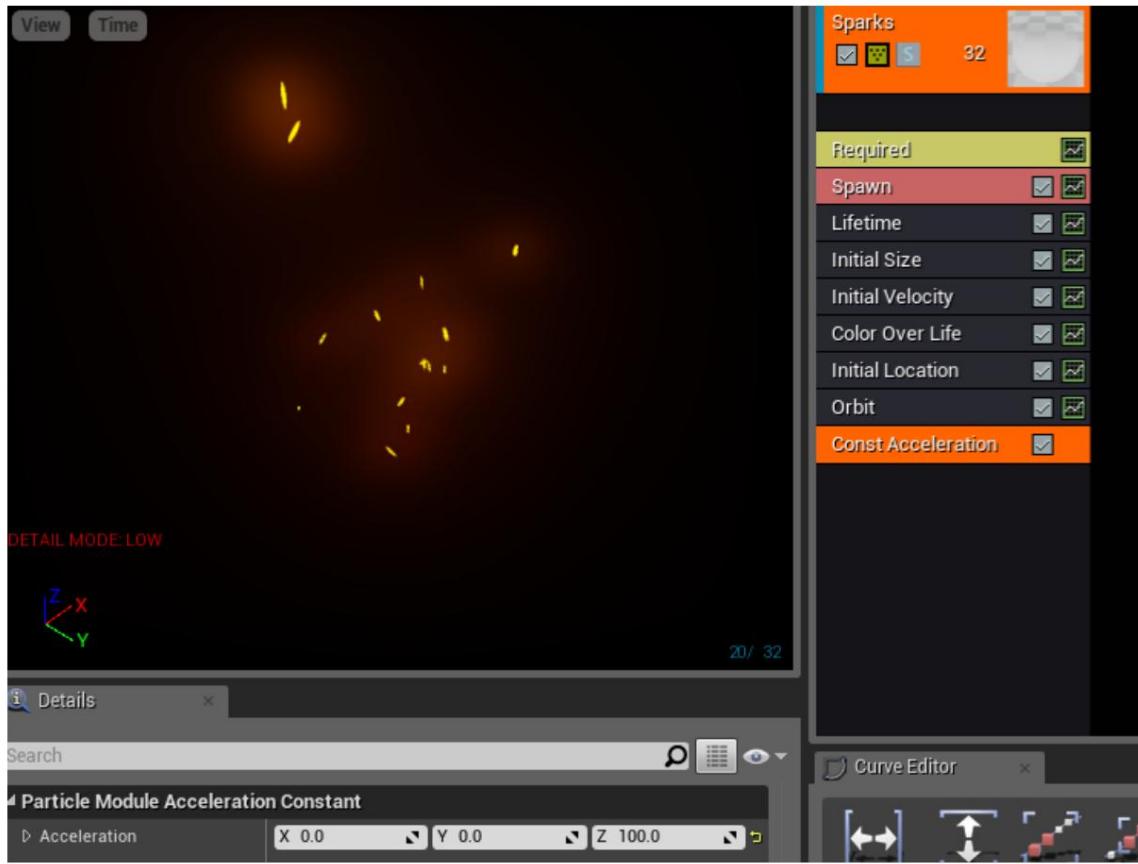


图 149. 应用轨道和加速模块后的粒子系统捕获

持续的。

来源：自制。

最后，我们将添加一个 Light 模块，以便每个粒子都有一个光源。没有需要更改参数。现在我们有了粒子效果。现在我只是我们拖动舞台来查看结果。



图 150. 游戏中粒子系统的捕获。

来源:自制。

5.2.3.4。添加照明。

在游戏中,决定使用引擎提供的两种类型的照明,点光源和光源。

点光源是放置在地图周围的点光源,可以为某些资产提供更大的照明,并帮助玩家了解他们可以与某些资产进行交互

NPC或触发器,光源已被用作关卡的全局光源,保持所有的舞台以可接受的光强度点亮。

为了正确执行光照计算,一个名为体积光质量,它计算给定封闭环境中的所有光子。

使用的灯没有特殊的修改,在一些强度已经修改产生更多的光,而在其他情况下,默认强度被认为是理想的。

它的添加方式就像游戏中的所有演员一样,通过拖动添加到舞台上使用鼠标,或者已添加为某些触发器或 NPC 蓝图的组件。

5.2.4。游戏音效。

虚幻引擎 4 中的游戏声音以相当简单的方式处理。

为了能够在游戏中播放声音,只需要:

1. 将音频导入项目。虚幻引擎仅支持 .WAV 格式的音频,导入时会转换为声波。对于这个项目,Audacity 程序已经完成了从其他格式到 .WAV 的转换。
2. 调用一个音频播放节点,很可能是Play Sound at Location,播放声音、播放 (音频组件)等。

指示我们希望音频循环播放直到我们命令它停止播放的方法是在相同导入音频的属性中定义的,在一个称为循环的复选框中,如果我们希望它不断播放,我们必须调用它。

5.2.5. 用户界面的实现

用户界面部分包括菜单实现、HUD 实现和库存。

为了在虚幻引擎中创建用户界面,我们使用了虚幻动态图形 (UMG),其核心是小部件。小部件是一系列预定义的功能 (如按钮、滑块、进度条等),可用于创建

我们的接口。这些控件在称为控件蓝图的特殊蓝图类型中进行编辑,其中包含两个用于创建界面的选项卡:

设计器选项卡,它是界面的可视层。

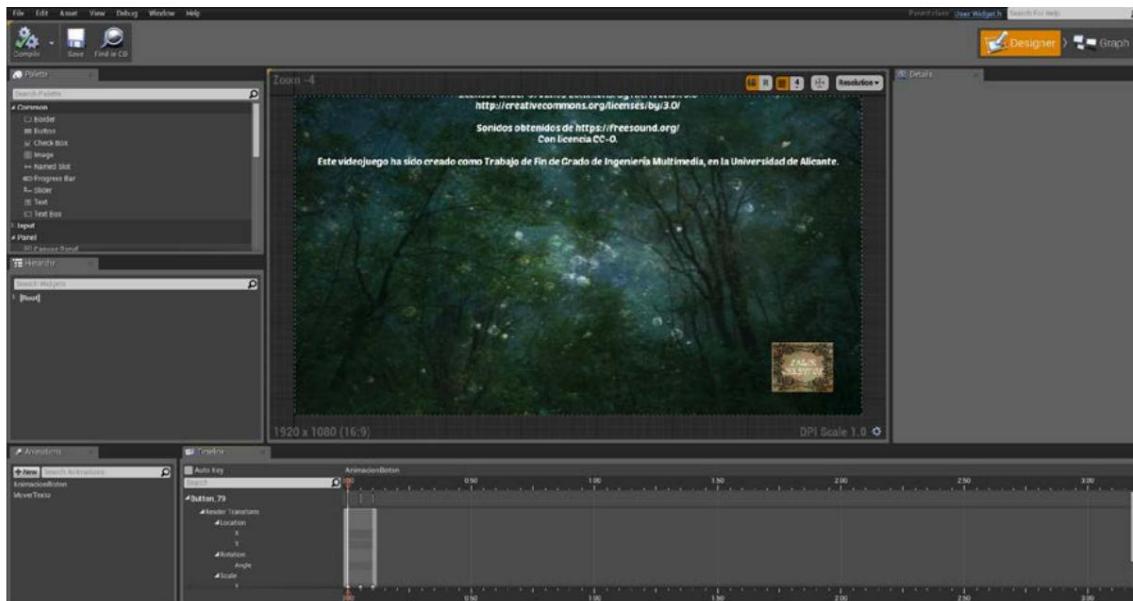


图 151. 蓝图小部件的 Designer 屏幕截图。

来源:自制。

图形选项卡 (图形),它将为使用的小部件提供功能。

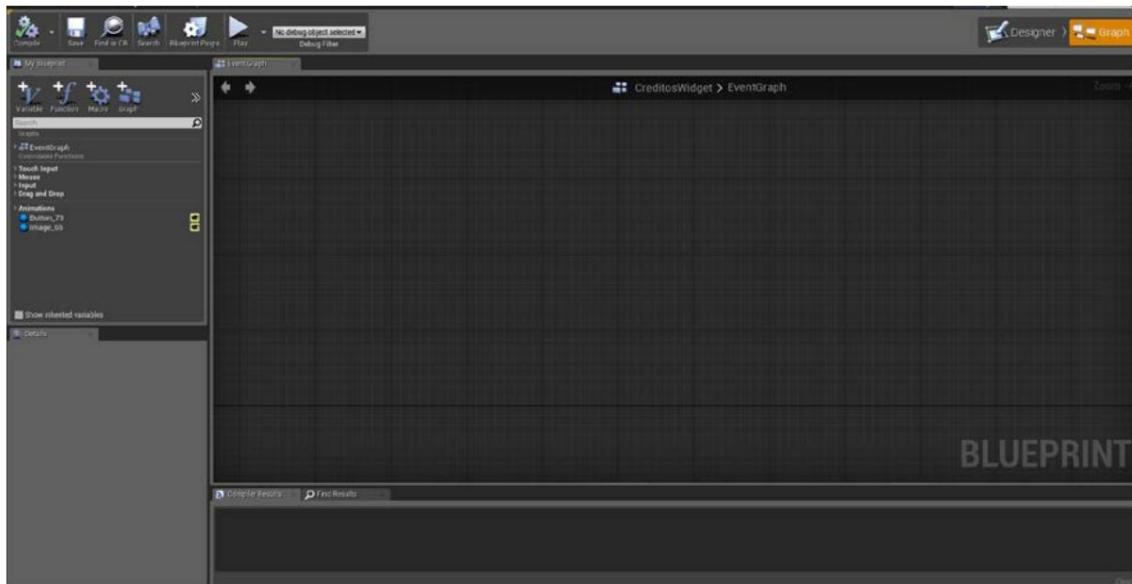


图 152. Blueprint Widget 的 Graph 选项卡的屏幕截图。

来源:自制。

换句话说,使用 Designer,我们可以在画布上添加元素、创建动画和自定义通用元素,并通过 Graph 为它们提供逻辑,例如,当按下新菜单按钮时允许加载关卡。离开。

除了上述之外,就像在其他视频游戏引擎(如 Unity)中工作的Widget Blueprint 中元素的定位一样,我们有一些 Anchors,以便能够根据屏幕类型随意放置我们的菜单元素。

鉴于在我们的特定情况下,我们不会找到不是 16:9 或 4:3 格式的屏幕,我们选择在屏幕上制作相对锚点。即无论是 800x600 的屏幕,还是 1920x1080 分辨率的屏幕,屏幕上的元素都会在屏幕边框内始终占据相同的位置,进行缩放。

5.2.5.1. 使用 Widget 蓝图创建用户界面的管道

无论是菜单、HUD 还是 Inventory,使用每个 UI 的基本管道都是相同的。基本上由以下几部分组成。

1. 创建蓝图小部件。
2. 在现有蓝图中,从您的事件图表中创建上一步中创建的 Widget 类的 Widget。
3. 创建一个变量来存储创建的Widget。最快的方法,虽然不是唯一的,是从创建的 Widget 中,将创建的对象作为输出参数,拖过它直到出现连接线,然后使用提升为变量,我们为它创建一个变量.我们将变量保存为

更大的安慰,因为如果不保存,每次要访问这个对象时,都必须从创建的 Widget 的节点开始画一条线。

4.当你想将Widget添加到游戏画面时,将我们在步骤3中创建的变量放置为Get,并通过它调用Add to Viewport 节点,将显示Interface Widget。

5.另一方面,当你想从游戏画面中移除 Widget 时,和上一步一样,我们在步骤 3 中创建的变量被放置为 Get,通过它,Remove 节点被称为 From Parent。

5.2.5.2。游戏内使用

正如游戏设计文档中所反映的,我们为 Arbennyg 构建了多个游戏菜单 (主菜单、选项、积分、确认、暂停、帮助) 和一个 HUD 界面,其中包括角色状态栏 (生命、能量和按键) 获得),作为库存。

需要注意的是,我们找到菜单的地图是主菜单地图,玩家的交互完全与 UI 交互,游戏地图,玩家有 HUD 和暂停菜单,以及其他 3D 菜单。与舞台元素交互时显示。

此外,每个按钮都提供了缩放动画,当它们被按下时,按钮在横坐标和纵坐标轴上都缩放 0.8,然后恢复正常 (缩放回 1)。动画在设计器中创建,并通过图形分配给每个按钮,当按下按钮时,我们创建一个播放动画节点,我们将创建的动画作为输入参数提供给该节点。此节点后跟一个延迟节点,延迟按钮后的任何其他功能,以便可以看到动画。

这些按钮在按下时以及光标放在它们上时也会发出声音,尽管游戏的这个逻辑方面只是设计师的问题,因为从图表中不需要添加任何功能。

接下来,我们继续讨论在创建不同用户界面时被认为相关的细节点。

5.2.5.3。主菜单

主菜单允许访问游戏关卡,以及在菜单之间移动和退出游戏。

突出显示的功能对应于对游戏关卡的访问,以及如何退出游戏。

要访问游戏关卡,必须在按下新游戏或继续游戏按钮时调用 Open Level 节点,我们在编辑器中指明已为关卡指定的名称。

另一方面,退出游戏是通过执行控制台命令来实现的,为此必须调用 Execute Console Command 节点,并在调用的命令中写入 quit。

5.2.5.4. 选项菜单

至于选项菜单,您可以通过它配置观看游戏的分辨率(默认为 1280x720)及其一般音量。

5.2.5.4.1. 修改屏幕分辨率。

对于分辨率,已经建立了 6 种不同的屏幕配置,3 种为 4:3 格式,
另外 3 个用于 16:9 格式。

修改屏幕分辨率,就像退出游戏一样,是使用控制台命令完成的。在这种情况下,使用 r.setRes 后跟所需的 WidthxHeight 分辨率(包括 x),例如 800x600,并且可选地在末尾添加 w 以指示我们是否希望屏幕处于窗口模式。出于最终美学的原因,已决定仅全屏运行游戏。

需要补充的是,虚幻引擎4的4.6.1版本,用于本项目直接兼容Oculus Rift DK1(以后版本会丢失),在更改分辨率时有一个bug,分配选择所有屏幕分辨率。经过多次亲身检查,找到了一个解决办法,就是先把分辨率设置为窗口模式,然后再设置全屏分辨率。

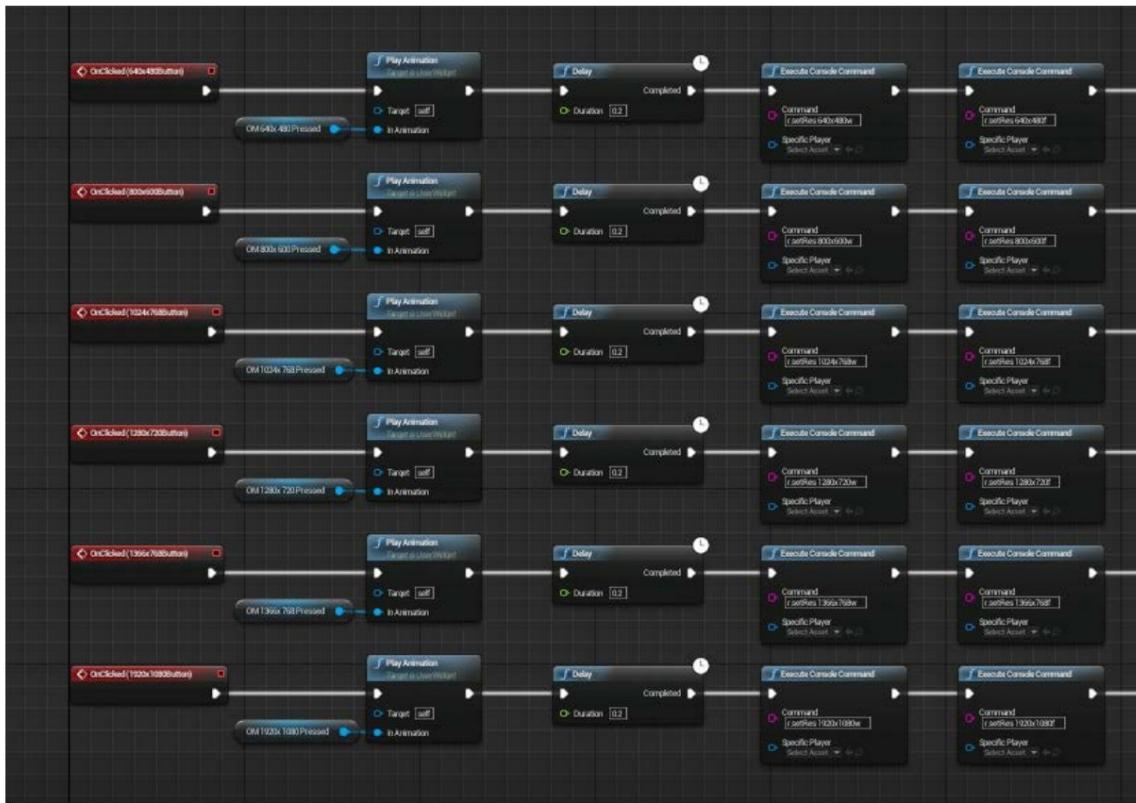


图 153. 更改屏幕分辨率的逻辑截图。

来源:自制。

对于可用性问题,一旦选择了分辨率,选择它的按钮就会被禁用。按钮的停用是通过 Set Is Enabled 节点完成的,要停用的按钮作为输入参数传递到该节点。

5.2.5.4.2. 修改整体游戏音量。

通过音频层次结构可以轻松修改游戏的总体音量。因此,创建了一个通用音频蓝图类,称为 BP_Master_Sound,以及其他三个类,即 UI、SFX 和 Music。除此之外,还使用了 Victory 插件,它为引擎添加了额外的功能,我们可以从 Unreal 网站免费下载。

通过 BP_Master_Sound 类,创建了三个子变量,也就是上一段中提到的三个,并且,在这个父类的图中,节点被链接以表明它们是子类。

当我们导入音频时,我们所做的是指明它所属的声音类,默认设置为 BP_Master_Sound。

因此,如果我们修改 BP_Master_Sound 类的音量,子类的音量也会被修改。此外,这允许在将来的更改中添加更多个性化的音频修改选项,例如允许单独修改每种类型的音量,并且这种分层结构允许创建新的子子类。对于这个初始项目,尽管是分层的,但它是

无论它所属的类的类型如何,因为音量变化适用于所有,但在可能的扩展中,此细节有助于添加任何音频功能。

最后,为了结束选项,在逻辑级别,我们使用一个值从 0 到 1 的滑块。在其 OnChange 事件中,我们调用以“Victory Sound Volume Change”来更新父音频类的一般值,并且另一方面,关于滑块的默认值,我们为此属性创建一个绑定函数,这将允许我们将值初始化为通用值,进而将其控制在逻辑级别。

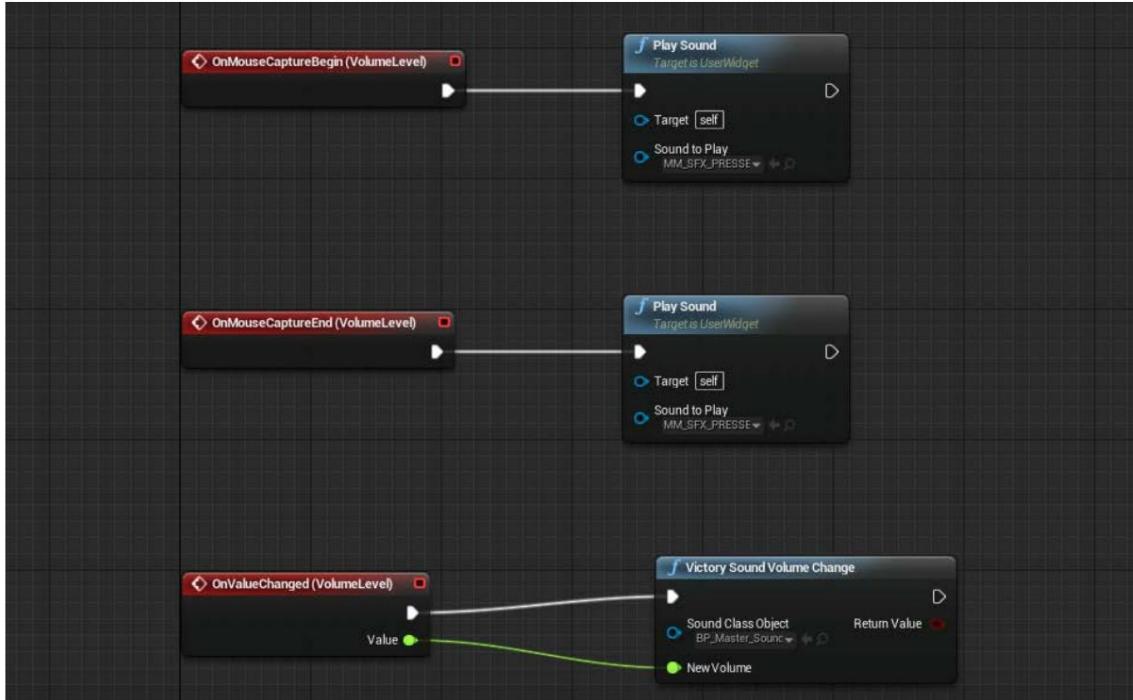


图 154. 修改游戏总音量的逻辑捕获,以及音频播放示例。

来源:自制。

5.2.5.5.暂停菜单

暂停菜单中要考虑的功能细节严格来说是暂停游戏的可能性。

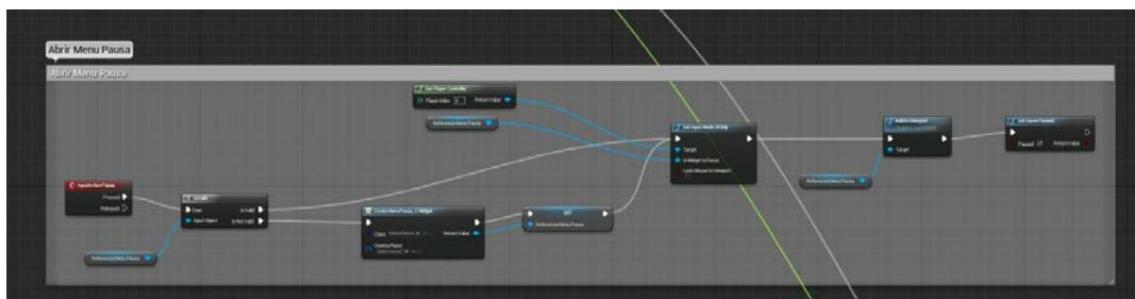


图 155. 暂停逻辑的捕获。

在虚幻引擎 4 中暂停游戏,有一个特殊的节点,这个节点是 Set Game Paused,它有一个逻辑变量,我们设置为真或假,取决于是否

我们想暂停或恢复游戏。虚幻引擎自动处理暂停游戏。

5.2.5.6. 平视显示器

HUD 突出显示的功能是它始终与我们角色的蓝图进行通信，因此，例如，如果他失去生命，HUD 必须在生命条中反映该损失。或者如果玩家捡起一个物品，这个物品应该出现在物品栏中（当玩家打开它时）。

为了实现两个蓝图之间的这种直接通信，我们选择从角色 Arbennig 的蓝图创建 HUD，以便始终保存对 HUD 的引用。此外，从 HUD 中，我们使用 Get Player Character 节点获取对角色的引用（输入参数 player index 设置为 0，因为我们只有一个），然后，我们将其转换为角色的特定类(Arbennig_Character)，节点 Cast 到 Arbennig_Character。最后，我们将这个转换分配给一个我们可以随时轻松使用的变量，从转换结果对象的输出中快速提升为变量。

一旦蓝图之间的沟通渠道建立起来，那些你想修改的值就被“绑定”了，这样我们的变量就一直处于等待修改状态，因此我们的 HUD 可以反映玩家发生的任何变化，比如失去生命或能量，或拿起钥匙。这些绑定函数从 Blueprint Widget Designer 分配，并从图表中给出逻辑行为。

例如，对于键，在 HUD 中创建了 8 个 ESlate Visibility 类型的变量（UE4 用来对元素执行可见性更改的数据结构类型），并在角色 Arbannig 中创建了 8 个 bool 类型的变量。我们在绑定函数中所做的只是检查分支是否有任何键碰巧为真，如果是，它的可见性将从隐藏变为可见。这样，如果玩家获得了密钥，HUD 将自动侦听该更改，并将继续使获得的密钥可见。

对于能量和健康，我们有进度条，而不是绑定可见性，我们绑定进度条的填充百分比值。在播放器中有一个从 0 到 1 的标准化值，该值会被简单地读取，并且百分比会显示在条形图上。

5.2.5.6.1. 存货

在这种情况下，库存的额外功能是包含最多额外功能的 UI。

它的逻辑操作允许我们在游戏中，如果我们打开库存，它会显示在屏幕上，如果我们关闭它，它就会消失。

此外，库存由两部分组成，一个是采样器部分，其中可以看到 10 个最初的空盒子，其中的对象将在收集时出现，另一方面，一个动作部分，它将使如果我们选择一个出现在清单中的收集对象，它是可见的。此操作子菜单允许您执行三个操作，使用/交付项目，放下项目（将其放到舞台上）和取消操作（如果您不想做任何事情）。

与对象）。使用/交付（如果可以对所述对象执行此操作）和丢弃都意味着该对象从库存中消失。

清单由一组元素组成，因此，与键的情况不同，键已经有了它们的位置并且只有在它们可见或不可见时才会改变，在这种情况下，当一个对象被拾起时，它会被添加到数组中，当一个对象被使用或丢弃时，它会从数组中移除，并且数组中的对象被优化放置，因此在它们之间不会留下任何间隙，例如，一旦一个对象被丢弃。

此外，物品栏不仅必须与玩家互动，还必须让可拾取的元素调用物品栏，以便在拾取时更新物品栏。

另外，当我们接近一个可以拾取的物体时，它上面会出现一个文本，指出它的名字。

也就是说，为了创建库存，它已被使用，除了 Widget 中的一个洞
用于显示库存的 HUD 蓝图，另外两个 Widget 蓝图：

1. 物品栏每个插槽的部件蓝图。

在这个蓝图中，它就像一个带有图像的按钮，我们要做的主要是两件事，绑定将具有 Blueprint Widget 插槽的图像，这样当玩家拿起一个对象时，这个图像就会通过是对象的，另一方面，当玩家点击这个插槽时，它会调用一个事件调度程序，该事件调度程序将点击的插槽作为参数发送，并在 HUD 上激活一个事件，指示选择了什么槽来处理。

按钮最初设置为禁用，直到分配了非 null 的图像（使用 IsValid 节点，我们将 Widget 的图像变量传递给该节点），这意味着直到我们收集到一个对象，并放置在插槽中，我们将无法与之交互。由于它是动态工作的并且必须监听变化，我们绑定按钮的行为来启用它（Is Enabled?），当它有一个空图像时它将被设置为 true。

不

在设计级别，在设计器中，我们消除了出现在其中的画布，因为我们不需要它，我们添加了一个按钮，并在其中添加了一个图像。在逻辑层面，要绑定库存槽将拥有的图像，因为我们无法绑定图像属性，但我们可以绑定绘制它的画笔，通过绑定函数中的代码，我们创建一个 Texture2D 类型的变量（类型我们在其中存储库存项目的代表性图像的变量），并且我们通过调用 Make Brush from Texture 节点将此变量转换为 Brush。

2. 一个控件蓝图，用于显示出现在要拾取的对象上的文本。

这个蓝图和前一个一样，不需要画布，因为我们只需要一个文本。虽然，为了正确定位它，我们创建了一个水平框作为容器，并在其中添加了文本。我们想要居中显示的文本，当我们创建小部件时，在其事件构造中，我们使用节点

在 Viewport 中设置 Alignment, 我们把它放在中心, 也就是在 x 和 y 的 0.5 处。

在这里,我们在图表中的逻辑允许我们显示或不显示 Widget 的文本,除了将其定位在视口 (游戏屏幕)中,我们将通过获取要显示的 actor 的位置来实现文本,我们添加到那个位置,在 Y 方向上 100 个单位,这样它就不会看起来粘在对象上,而是稍微偏移了一点,我们根据玩家控制器在屏幕上的坐标中转换那个位置 (我们获得了玩家控制器,获取索引为 0 的播放器控制器),为此,使用播放器控制器节点,我们获得将世界位置转换为屏幕位置节点,我们将显示文本的位置作为输入传递给该节点,并返回屏幕上的位置。为了确定文本的可见性,要转换为屏幕坐标的节点除了屏幕上的位置外,还返回一个布尔值,指示是否可以执行转换,即,例如,如果玩家朝上看,或者背着看,2D坐标将不会被分配,根据这是否发生,我们创建一个布尔选择,根据它的值,我们可以确定Widget的可见性 (ESlate 可见性)。

以及与 Widget 蓝图不同类型的三个蓝图:

1. 蓝图界面,它允许我们将 HUD 与我们可以收集的元素进行通信。在这个特殊的蓝图中创建了两个函数,使用函数和放置函数。对于 drop 函数,我们在它的图中添加了一个 actor 变量,它对应于我们将要删除的 actor。

2. 一个结构类型的蓝图,它基本上是不同变量的集合,我们将使用它作为库存中每个插槽的结构。在这个蓝图中,我们添加了 4 个变量,第一个是 actor 类型的,它是我们要收集的物品,第二个是 Texture2D 类型的,它是代表库存中物品的图像,以及另外两个变量,文本类型,一种用于将出现在项目上的浮动文本

当我们拿起它时,另一个用于将出现在使用字段中的文本,如果它来自任务任务,它可以是交付物品,或者如果它是用于治疗的物品,则可以使用。我们将此蓝图称为库存结构。

3. 一个或多个 Actor 类型的蓝图,这将是从舞台上收集的不同元素。这个 actor 必须包括一个网格和一个碰撞触发器来检测玩家,作为变量,一个布尔值来知道玩家是否在触发器内 (因此,元素信息消息将被显示),一个库存结构类型的变量 (之前创建的结构类型蓝图),结构的所有组件都填写在这个变量的属性选项卡中,除了 actor,我们用代码表示,最后,这个蓝图,我们添加变量 (我们必须放置作为公共或可编辑的,以便它们可以从编辑器中修改) 在使用时项目将做什么,以防我们需要某些东西,例如,如果角色的生命值增加,我们为其添加一个浮点变量。

解释了上述内容后,在 HUD 中,我们在设计器中添加了两个垂直框,一个包含库存本身,另一个包含从库存中选择项目时将显示的操作子菜单。在库存的垂直框中,我们添加一个文本块 (向用户表明他看到的是库存) 和一个统一网格面板,以及什么

我们所做的是用 10 个 Inventory Slot 填充它（它出现在 User Created 选项卡中，因为我们自己创建了它）。在动作子菜单的垂直框中，我们添加了一个文本块以向玩家指示他可以执行其中显示的动作之一，以及 3 个按钮，使用按钮、释放按钮和取消按钮（执行没有什么）。

一旦创建了上述所有内容，我们要做的是：

1. 在继承我们创建的接口的 HUD 类中，我们创建一个 RefreshInventory 事件，我们在其中创建一个包含 10 个库存槽类型变量的数组。我们使用 foreach 循环遍历数组，检查库存是否已满（如果已满，我们什么也不做），如果没有，我们分配 Pickup 图像，然后，我们将节点分配给每个库存槽 库存中要单击的项目的自定义事件，这将禁用库存，以便出现操作菜单。

此外，在这个类中，我们为操作菜单按钮提供功能，使：按下取消操作按钮时，库存会重新激活并隐藏

操作菜单。

湾。当按下放置物品按钮时，让我们通过接口向我们要放置的蓝图发送一条消息，并在从库存中发送该消息后通过删除该物品来更新库存，并调用自定义操作完成事件，以便重新激活库存。

C. 当按下使用物品按钮时，会通过蓝图接口向我们要使用的具体蓝图发送消息，然后更新库存并调用动作完成事件。

2. 在我们用来创建可收藏元素的类中，我们所做的就是继承，就像在 HUD 中一样，我们必须继承已经创建的动作蓝图界面，创建显示在对象上的 widget，我们可以使用它交互，并分配要显示的文本，然后分配一个 PickupItem 自定义 Event，它将检查玩家是否在范围内拾取物品，如果是，则拾取它，将其添加到库存中，并更新它。

此外，在此类中实现了使用操作，这取决于对象，可能具有一种功能或另一种功能。

3. 最后，在继承 Action 接口的 Character 类中，我们要做的是控制玩家是否按下库存键来显示它，并在游戏中启用鼠标控制。

另一方面，如果玩家与任何可以拾取的物品进行交互，则会调用拾取的收藏品类中的 pickupItem 事件。

最后，对于我们的角色，我们必须实现 drop 动作事件，从该事件中我们接收到一个变量，其中包含我们要丢弃的对象，然后，我们重新激活它的碰撞，让它再次可见，然后我们改变它的位置到一个放置位置。

5.2.5.6.2. 振铃测试计时器

对于这个定时器的实现及其对应的逻辑根据测试，
所做的一方面是在它的刻度功能中具有时间计数器的更新，
已决定以添加每个瞬间的千分之一值的方式计算时间，
当它达到值 1 时，从剩余时间中减去一秒。

另一方面，在 tick 函数中，我们也检查剩余时间是否到了，或者是否
剩余和获得的戒指（获得的戒指从 LevelBlueprint 更新），
match，以确定挑战是否结束。我们将调用自定义事件
在图表中将负责标记挑战已经结束（以及是否成功），以便从关卡蓝图中我们可以得到证明。此外，玩家被传送到挑战岛，这个
蓝图部件的箱和时间消失了。

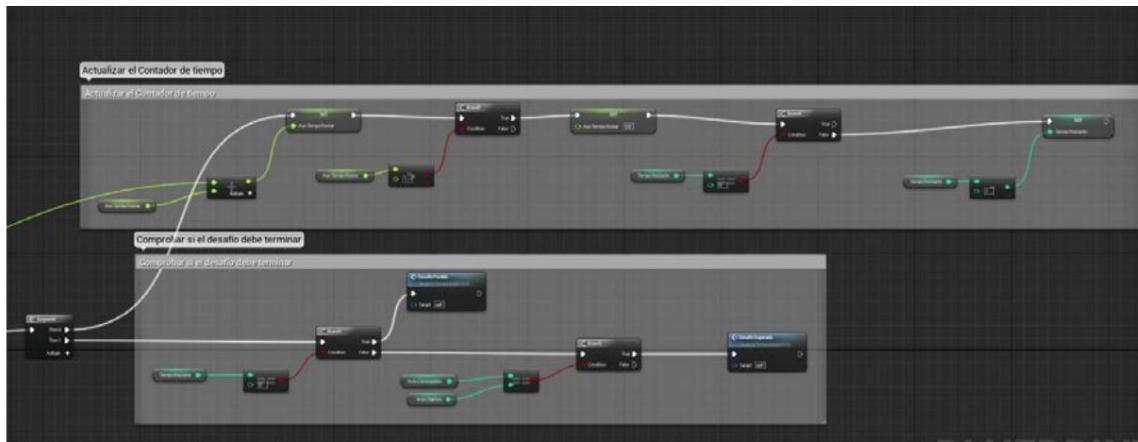


图 156.捕获计时器的游戏逻辑。

来源：自制。

5.2.5.6.3。序幕、尾声和演员表屏幕。

序幕、尾声和演员表屏幕的实现包含相同的逻辑
游戏，没有超过准时的变化来决定这些屏幕一次指向的位置
它的功能已经结束。

这些屏幕在设计器中有一个背景图像、显示在屏幕下方的动画文本、一直向上滚动，以及一个跳过“过场动画”的按钮。

每个屏幕都处于不同的游戏关卡中，并在完成时加载其他关卡。
向上移动文本或按下按钮。

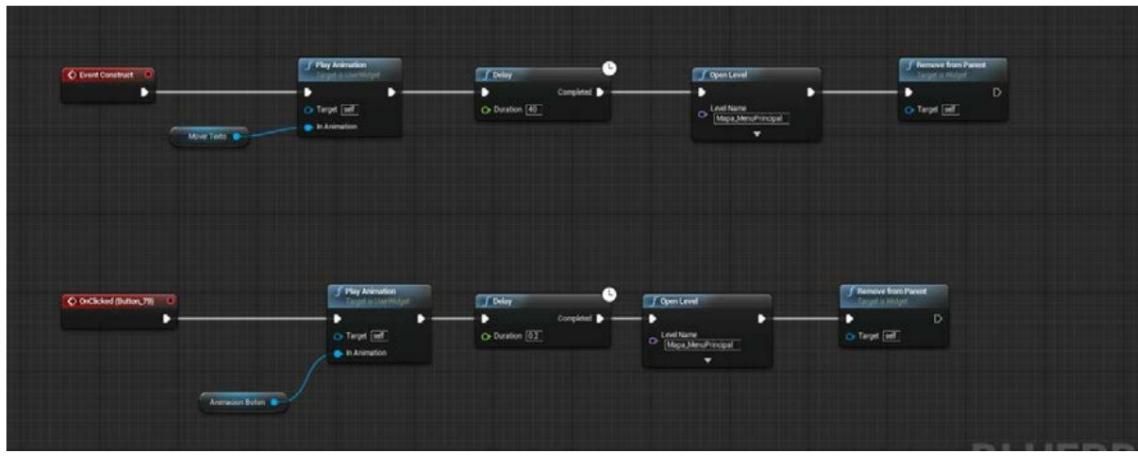


图 157. 学分类逻辑的捕获（序言和尾声相同）。

来源:自制。

当文本完全上升时,级别发生变化的逻辑是检查

动画时间轴中的持续时间,无论它是什么持续时间,我们都将它放在里面的延迟中
事件构造的,在播放文本的动画之后,以及当它结束时
延迟,打开另一个级别。

另一方面,按钮的逻辑是,当您按下它时,它指向的级别打开。

5.2.5.6.4。挑战通过和失败的屏幕,保存的游戏,你不能使用该项目。

这是一系列 Widget 蓝图,其设计器中有一条消息。当它们被创建时,它们的内部逻辑在它们的事件构造中有一个延
迟,当几秒钟过去时,
导致他们自毁。

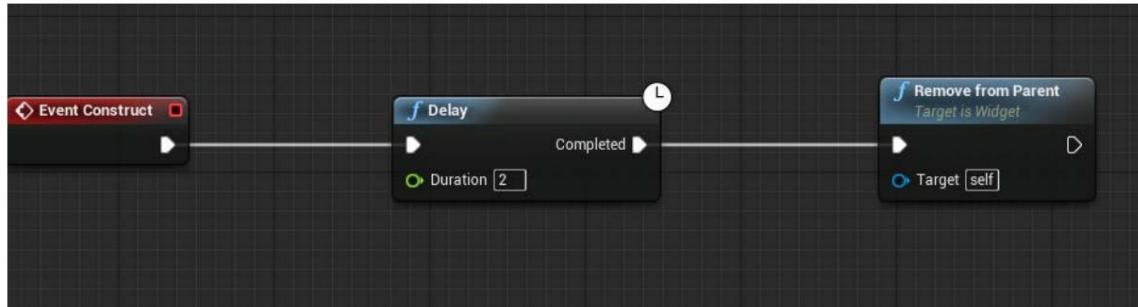


图 158. 具有指示功能的蓝图小部件的屏幕截图。

来源:自制。

5.2.6。游戏机制的实现。

在这个项目中,所有游戏机制都以类似的方式工作。我们有考虑到每个不同的机制将是一个不同的蓝图,我们可以拖动到舞台(或动态生成)。

在任何情况下,对于每一种机制,本质上任何一种机制都是必需的。它们是触发它被编程的功能的原因,在这个项目中,所有机制都通过玩家碰撞的触发器来激活它们的功能。另外,可以为这些触发器附带一个网格,使玩家能够识别出他所面对的元素不是简单的装饰。

也就是说,项目中包含的每个机械师的最详细和具体的操作如下所示:

5.2.6.1.传送器

传送器位于岛屿更南端的森林中
游戏地图,并在地图以北的迷宫中。

这个机制的逻辑是将角色的演员移动到地图上的另一个点。

已决定通过创建两个蓝图来实现此功能。蓝图
传送器,如果玩家与之碰撞,他们将传送,以及目的地蓝图
传送,如果玩家与之前的蓝图发生碰撞,就会出现在传送中。通过
传送器蓝图,我们指出传送器应该传送到地图上的哪个蓝图actor。
播放器。

通过蓝图描述它的功能,一旦两者都被创建为类型的蓝图
演员,我们需要做的,一方面,在传送器蓝图中:

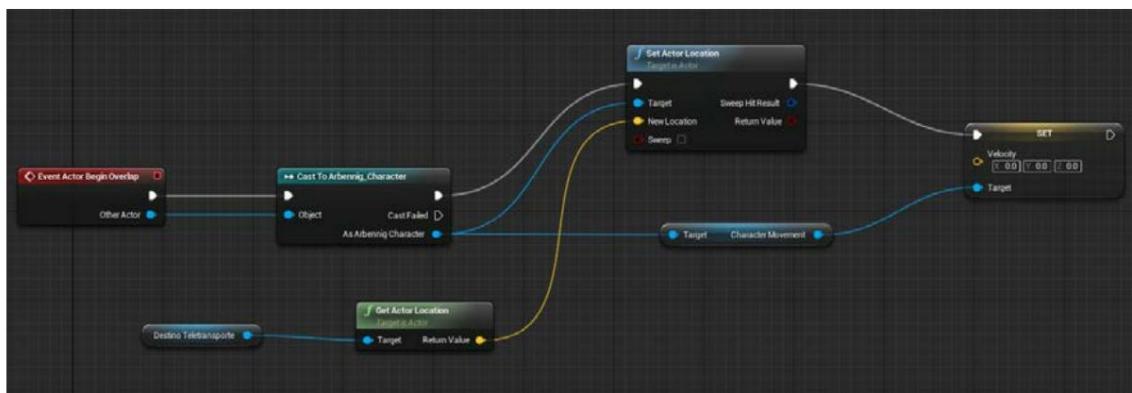


图 159. 传送类逻辑的蓝图捕获。

来源:自制。

1. 添加玩家与传送器触发器交互时的事件。
2. 从这个事件中,我们得到碰撞的actor,并将它投射到
Arbenneg_Character,也就是我们的角色。
3. 从这个演员表中,我们得到我们的角色,我们点击变量 pin
我们调用 Set Actor Location 节点来设置新位置。
4. 为了得到目标位置,我们创建目标类的蓝图变量
通过变量详细信息选项卡传送 (另一个蓝图),并通过选中可编辑复选框使其可编辑 (当我们制作可编辑变量时,
可以从关卡编辑器中访问它)能够通过编辑器

初始化它以指示传送对应的目的地。这个变量

我们拖到图表上,表示我们想以 Get 的形式获取它,并从中拖动它的 pin 以获取 Get Actor Location 节
点 (这将为我们提供 Destination 位置),并且

最后,我们将这个节点提供的值拖到

设置我们在步骤 3 中调用的 Actor Location 节点。

5. 再次从我们的字符变量中拖动,我们得到它的字符

运动,我们称之为 Set Velocity,这样如果玩家在

移动,它的速度变为0。

在传送目的地蓝图中,出于游戏性原因:

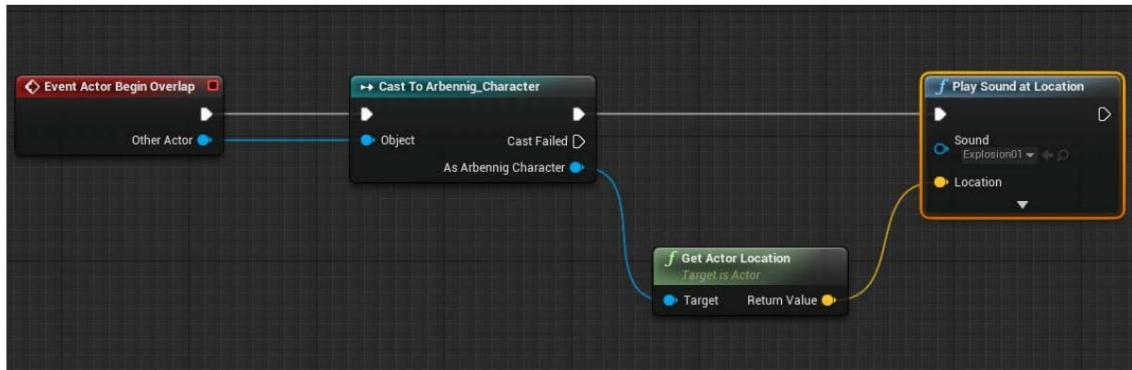


图 160. Teleport Destination 蓝图的捕获。

来源:自制。

1. 添加玩家与传送器触发器交互时的事件 (当
玩家已经传送到这个位置,它将激活)。
2. 从这个事件中,我们得到碰撞的actor,并将它投射到
Arbenneg_Character,也就是我们的角色。
3. 我们调用 Play Sound at Location 节点,并从我们的内容中选择一种声音
浏览器并设置播放器使用获取节点的位置
演员位置取自演员表。

5.2.6.2。等待

当玩家在说话后进入游戏中的飞行挑战之一时会出现戒指

使用寺庙图腾 4 或 7 并接受任务。

玩家会看到一个计时器出现,以及在游戏中获得的戒指的信息文本。

屏幕上,所有的箍都被激活并变得可见,你必须通过其中的每一个

在时间到之前。当球员通过篮球筐时,球员执行轻微的

动画(旋转得更快),几秒钟后,它消失了。另外,如果时间到了

0,箍消失,如果玩家愿意,必须恢复以供下次尝试。

重试。此外,当飞行挑战结束时,玩家会被传送到该位置

对应的图腾所在的位置。

环形逻辑使用 4 个主要蓝图:

1.我们将要创建的 Widget Blueprint 类型的蓝图,以便它在界面中重叠

给定的时刻,因此HUD不会超载逻辑。在这里我们照顾

保持剩余时间计数器和收集的响铃相对于

其余的部分。除此之外,通过这个蓝图,我们将检查挑战是否

结束与否,成功与否,并据此指出,以便在

关卡的蓝图将继续给予玩家符文与否。终于通过了

这个蓝图决定将玩家传送到

找到图腾

2.箍蓝图。这个蓝图,已经建立了两个版本,还有一个

简单,更复杂的粒子效果,位于舞台上

以一定的速度旋转。当玩家与里面的触发器发生碰撞时,

它开始旋转得更快,两秒钟后它消失了。此外,它重置

最大化玩家的能量,并在一个变量中记录它已经被遍历,这样

你可以在关卡的蓝图中进行。

3.人物蓝图。它通过不同的蓝图访问角色以

修改其符文、能量、位置和状态变量。

4.游戏关卡蓝图。有了这个蓝图,我们将开始挑战篮球,

从而阻止任何其他挑战的完成,并重置它们的值以防万一

不甘示弱。此外,我们将向 Timer Widget Blueprint 发送信息,告诉它经过的更新的圈数,最后,还有

我们将检查挑战是否应该结束,给玩家符文,并重置游戏,以便可以完成其他挑战。

接下来,我们继续在逻辑层面上解释最相关的步骤。

制作每个已创建的蓝图 (因此,不是已经创建的角色以前,并且没有添加任何代码)。

蓝图计时器小部件。

在这个蓝图中,我们有一个用于计时器的文本小部件,另一个用于获得的戒指关于总数,还有几个显示挑战完成与否的消息。这些文本包括一个背景图像,以更加突出。

我们所做的就是将计时器文本的内容与获得的戒指进行绑定。我们将此绑定与一些整数类型的变量相关联,我们将首先将其转换为字符串以便能够附加(它允许我们连接字符串),因此添加剩余秒数的“s”或“/”和数字

总戒指。另一方面,我们绑定通过或失败的挑战文本的可见性,将它们分配给 ESlate Visibility 类型的变量,以便在需要显示它时我们可以控制它。

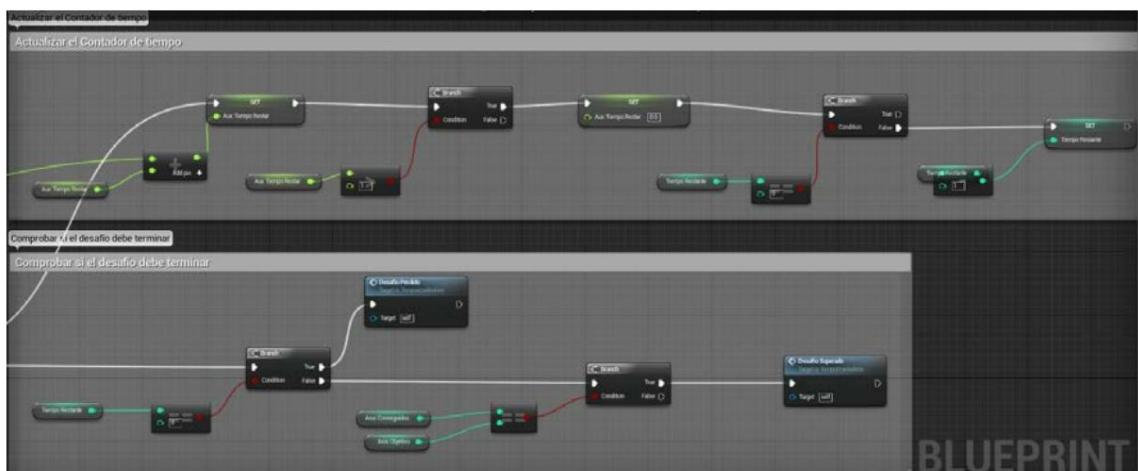


图 161. 捕获 Blueprint Widget 的图形逻辑

来源:自制。

至于这个蓝图的主图,我们从一个刻度函数开始,它将去在挑战未完成时调用两个引脚的序列。在这两个引脚上,第一个负责更新时间计数器 (取决于增量时间,它添加对一个辅助变量,当值达到 1 时,从计数器中减去一个单位,辅助变量重置为 0),第二个负责检查剩余时间是否为 0,以调用丢失挑战自定义事件,并检查是否没有时间用完,如果数量达到的锚数等于目标锚数,如果是这样,将调用另一个自定义事件,但克服了挑战。

挑战失败和克服的自定义事件,他们做的第一件事就是改变可见
挑战文本传递,其图像背景,在此之后,将挑战变量设置为 true
完成,以及指示挑战是否成功完成的真或假变量。对此
接下来是 3 秒的延迟,以便玩家可以欣赏他通过的文本或
不是挑战。一旦延迟时间过去,角色的状态就会改变,无论
是 (可能是飞行) ,下降,因此它自动设置为行走 (似乎
我们已经被传送并摔倒了) ,最后我们从父级调用 remove 来移除蓝图小部件。

蓝图德尔阿罗。

在这个蓝图中,让我们记住,我们有两个 (虽然唯一的区别是一个有网格,另一个在触发器周围有粒子效果) ,我们要
做的是有一个
滴答功能,通过局部旋转不断转动箍,应用
将增量时间乘以您选择的值,在一个轴上创建一个旋转器,以及
使用 Add Local Rotation 将其应用于网格。在应用旋转之前,我们包括一个
检查玩家是否已经通过触发器,如果是这样,乘法
成为另一个更大的数字。

当玩家通过触发器时,它会触发 OnComponentBeginOverlap 事件,其中
我们检查触发器是否已经被越过,我们激活 bool 变量以便环旋转
更快,然后,我们为玩家补充能量 (让角色拥有
获取 Player Character 函数并转换 Arbannig Character) ,我们将变量设置为 true
表示环已被越过,我们会延迟几秒钟 (这样
环有 2 秒快速旋转) 。延迟之后,我们将指示的变量设置为 false
让箍快速旋转,我们用 Set Actor Hidden in Game 隐藏这个actor。

游戏关卡蓝图。

对于篮球的这一部分,我们要做的是使用开始播放事件,以及在这个
关卡蓝图,加上使用事件激活图腾挑战。

在开始游戏中,我们获得了对放置在舞台上的箍的引用,我们创建了一个
数组,然后我们将该数组返回给我们将在 tick 函数中访问的变量
反复。每个挑战的戒指必须排成不同的阵列。

另一方面,在 tick 函数中,我们执行一系列检查,包括
检查环挑战是否已被激活 (我们在接受挑战时激活它
图腾) ,以及挑战组件是否已经初始化。话虽如此,我们初始化

首先挑战组件（因为我们还没有初始化任何组件）,其中每个数组的环我们将其可见性从隐藏更改为可见,我们创建了蓝图小部件 timer 并将其分配给一个变量,以便以舒适的方式对其进行引用,并且我们添加到视口,然后访问这个小部件,我们将环初始化为 0 实现,我们通过制作环数组的长度来指示目标环,我们分配挑战时间,最后是传送目的地。在这个初始化部分的最后,我们将一个变量设置为 true,表示我们已经初始化了组件,并且我们从 tick 函数开始,我们将在其中更新获得的箍并检查挑战是否以及如何完成。

为了更新环,我们选择了对环数组进行 foreach 循环,这样对于每个元素,我们检查它的遍历变量是否为真,并且我们添加到获得的环变量。当循环结束时,我们访问变量从蓝图小部件获得的环,我们设置结果,最终重新启动为 0 我们获得的箍变量,在刻度函数中,将添加值不断地。

至于检查挑战是否结束,我们要做的是访问蓝图小部件并检查我们创建的变量 bool 是否表明挑战终止是真还是假。如果为真,我们会检查您的布尔挑战变量是否通过成功也是真或假,不管这个,我们将其设置为假表明我们的挑战是活跃的变量。

如果玩家成功完成挑战,我们访问 rune 4 变量来放置它为 true,存储在我们角色的角色中（我们使用 get player 函数访问它字符,并由此对 Arbennig_Character 进行强制转换）,最后,我们将 totem challenge enumerator variable to none,这样更多的挑战是可能的。

如果玩家没有成功完成挑战,变量恢复为 false 初始化组件,以便在您想重试时可以重新初始化它们挑战没有克服。此外,所有环的可见性都更改为隐藏（如果我们有克服挑战,它们都将被隐藏）,并且还执行数组的 foreach 循环对于箍,我们将表示已遍历箍的变量设置为 false。最后,我们将达到的环的变量重置为 0,这样当我们再次开始挑战时,交叉环的第一个值不是上一个游戏的值,我们已经确定了枚举变量,表明我们面临的挑战。

关于包含激活图腾挑战事件的逻辑,简单来说我们设置我们将要执行的挑战的枚举数（4 或 7）,并且我们设置为 true 篮球挑战触发变量。

对于与其他人的篮球挑战,程序完全相同。

5.2.6.3。 NPC对话和图腾

游戏中出现的对话会在玩家与任何 NPC 交互时显示
(城镇村民、树木和图腾)。它们由一个对话框组成,其中部分
顶部是来自 NPC 的消息,下方是可用的回复
的播放器。

通过这个系统,玩家和 NPC 可以进行多响应对话。在
这个项目,我们的角色可以和村里的NPC和树无限对话,
但是对于图腾,一旦挑战被接受并通过,图腾就不能说话了,因为
“他的灵魂被释放了。”

要实现这个对话系统,需要创建两个小部件 (一个
选项小部件和对话框屏幕小部件),三个结构 (对话框树,NPC响应,
和角色选项)和演员 (对话系统)。另外,你需要做几个
调用我们角色的蓝图。

所有这些结构和类如何组合在一起如下:

- 对话系统参与者包含对话屏幕小部件 (它只是
只是视觉上的东西,它将作为不同响应的容器,两者都来自
NPC as of the character) ,它又将动态包含
对 NPC 的响应 (由一个带有文本的按钮组成) 。
- 对话树结构包含 NPC 的响应结构,反过来,
包含角色的选项结构。
- 当我们与 NPC 或图腾互动时,我们会根据角色的蓝图
除了生成actor之外,还必须在用户界面上启用控件
我们想要展示的。

为了启用对 UI 的控制,我们创建了一个 InitMouse 函数,其中
我们创建对播放器控制器的引用,并从中设置模式
SetInputModeUIOnly,除了修改bool变量显示或不显示红色十字准线外
hud、显示光标和启用鼠标事件。

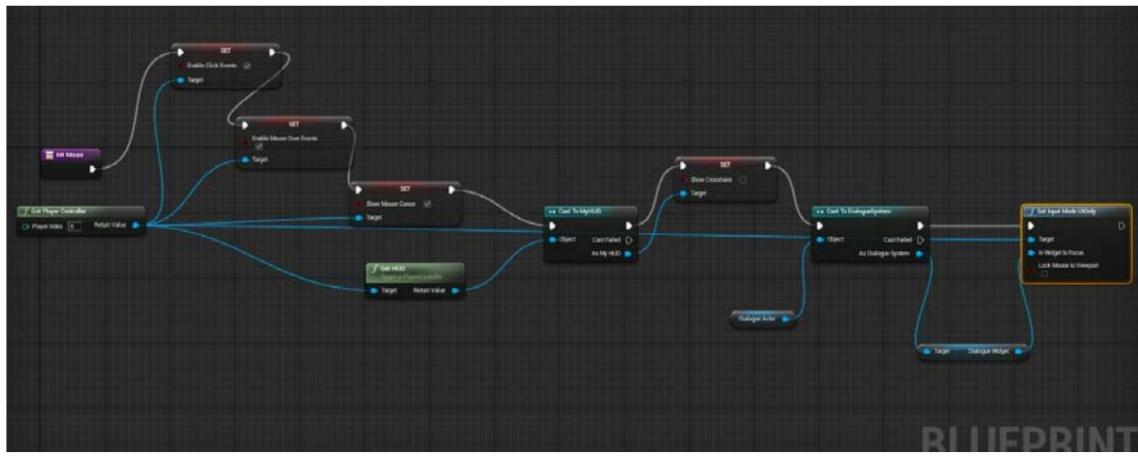


图 162. 捕获 initMouse 函数

来源:自制

另一方面,为了生成 actor,我们创建了另一个函数 SpawnDialogue,并在其中
我们简单地调用 SpawnActorFromClass 节点。作为类,我们选择 Blueprint 类
DialogueSystem (对话系统) ,最后我们保存这个创建的actor的变量。演员产生的位置无关紧要,因为当它被创建
时,小部件出现在视口中。每当您与任何 NPC 交谈时,您都可以像这样轻松访问它

通过我们的角色对对话演员。

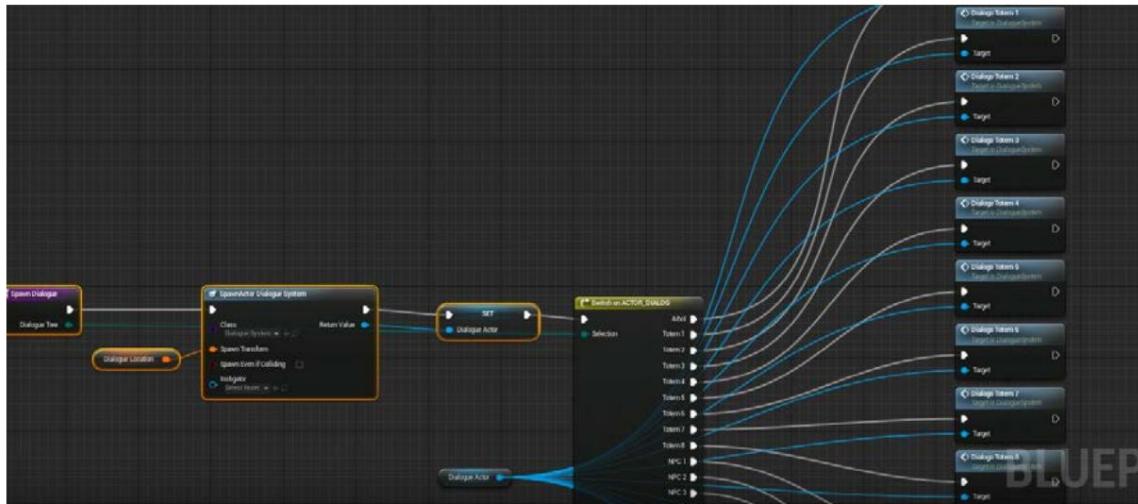


图 163. SpawnDialogue 函数的捕获

来源:自制。

至于结构,它们包含以下内容:

- 字符选项结构 (PC_Choices) 。包含 Reply (响应) 、LinkToNode
(链接到它指向的响应节点) 和 exitDialogue (布尔以确定是否有
此选项结束对话) 。

- NPC 响应结构 (NPC_Reply)。包含 NPC_Reply (来自 NPC 的回复) ,

和一组 PC_CHOICES。

- 对话树 (DialogueTree)的结构。包含一个对话变量,即

un array de NPC_Reply。

就 Widgets 而言,Dialog System Widget 里面没有任何逻辑。

它的类,我们这里只是有一个边框类型的容器,其中包含一个文本 (

来自 NPC 的响应)和另一个边界,其中又包含一个垂直框 (其中

玩家可用的答案将被动态添加) ,但这是必要的

表示文本和垂直框都是变量,并将它们公开,以便可以从外部编辑它们。在响应小部件中,我们必须将响应的文本设为公共变量。

按钮,并且,在逻辑层面上,我们所做的是从我们的角色中获取 DialogueActor,并且

调用一个我们将在这个actor中创建的函数,名为LoadDialogue,它接收为

LinkToNode (选项指向的节点)和 ExitConversation 参数,以及它将做什么

这个功能是每次我们按a时加载/重新加载游戏的对话系统

选项。

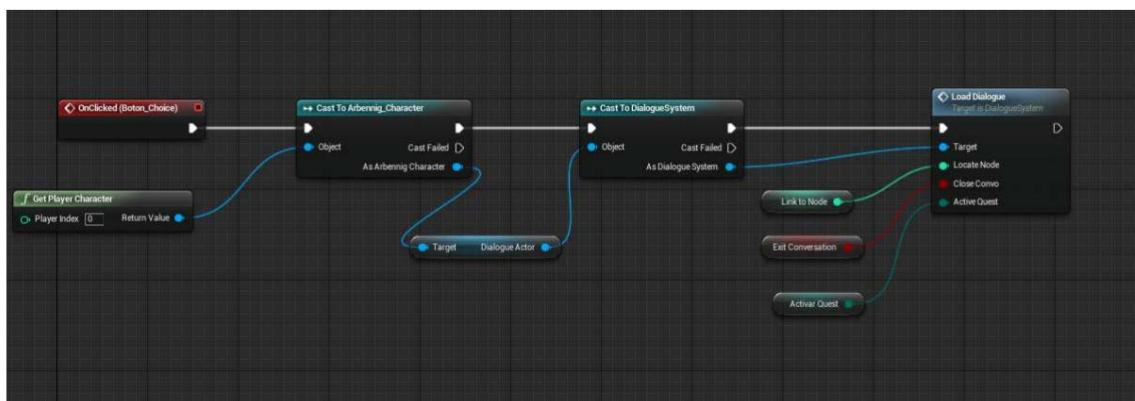


图 164.对话框选项小部件的逻辑捕获。

最后是 DialogueSystem 类。在这个类中,我们在主图中有逻辑,在两个

我们创建的函数,一个是LoadDialogue,已经提到过,另一个是StartConversation,

我们所做的就是调用 LoadDialogue 函数。

在类的主图中,我们创建类型数据结构的变量

每个不同对话的对话树和分配此值的一般对话树。除了

一方面,我们有一系列自定义事件,这取决于 NPC 或图腾

我们谈话将跳过一个或另一个,从而为谈话树结构分配一个值

或其他。另一方面,通过 begin play 事件,该事件将在

这个演员,我们创建一个 DialogueScreen 小部件,我们将其值存储在一个变量中,并且

我们通过 add to viewport 节点将它添加到视口,最后,我们调用函数

开始对话。

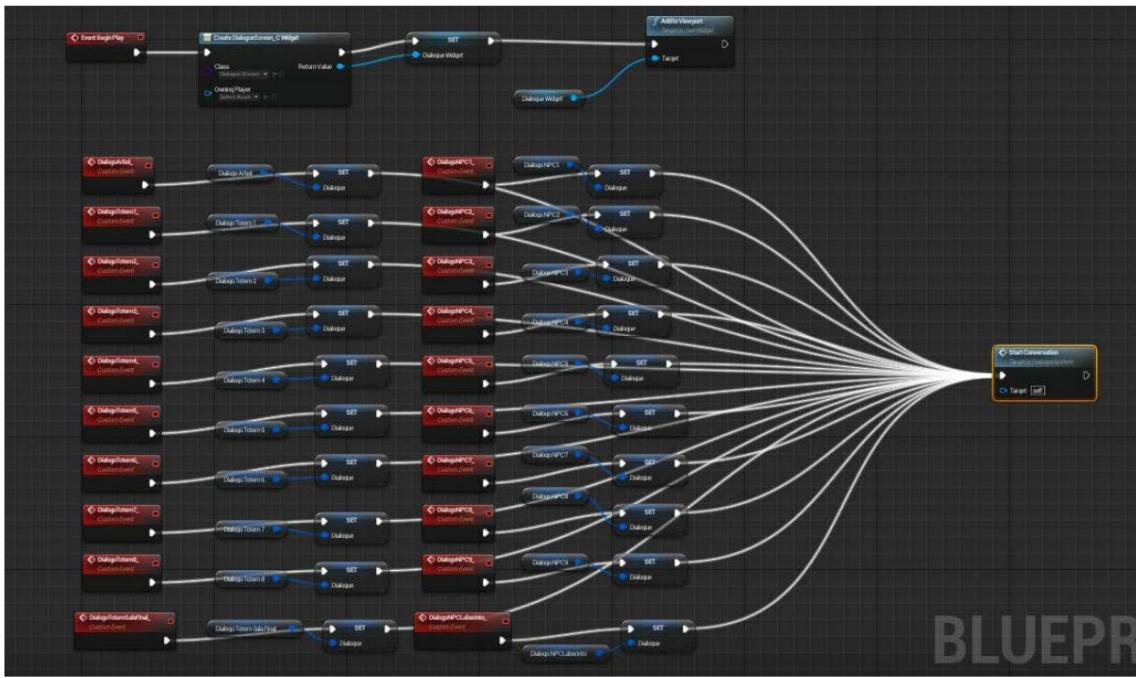


图 165. DialogueSystem 类的图形节点。

来源:自制

最后，在`loadDialogue`函数中，我们所做的是调用带有条件的分支

作为函数的输入参数开始的退出。如果属实，则意味着

对话结束,我们将调用 `DialogueScreen` Widget 的 `setVisibility`,另外

调用角色的自定义事件以重新激活控件。

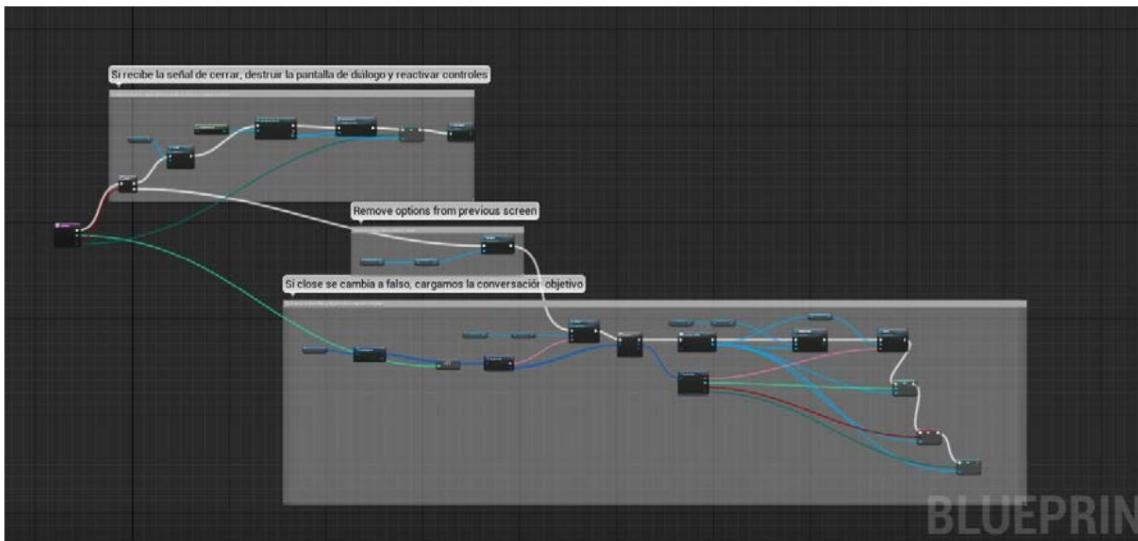


图 166. LoadDialogue 功能完成。

来源：自制。

如果分支为假,我们所做的就是加载 DialogueScreen 小部件中的所有内容
我们有,但首先,我们访问对话选项的垂直框容器,并且
我们用干净的孩子清洁它。为了正确加载对话,我们使用另一个
函数的输入参数,对应要读取的会话节点 (与
节点,例如,一个 NPC 响应,以及该响应的选项),以及
有了这个,使用我们在主图中分配的对话结构,基于
从我们与之交谈的NPC那里,我们得到了对话的node.of (我们称之为struct的get函数,它的索引就是我们想要的节
点的值)。之后,我们打破了这个对话节点,它是一个 NPC_Reply 的结构体,并用它来分配文本的值

来自 NPC 的响应 (我们为此使用 SetText 函数)。

最后,我们必须将玩家的响应选项添加到我们拥有的垂直框中
为此做好了准备,我们通过调用 PC_Choices 数组的 foreach 循环来做到这一点
我们有 (当我们破坏了结构 NPC_Reply 时,请记住它包含一个
NPC_Text 和 PC_Choices 数组),对于其中的每个元素,我们调用 Add Child 节点
Vertical Box,并从中分配正确的文本,即由
响应,以及 ExitConversation 的变量 bool 的值。

5.2.6.4。游戏结束时的大门。

游戏的末地之门机制包含两部分的功能,一个触发器
将通知该级别的班级门已准备好打开,并触发
门,其中包含通过带有时间线的动画,打开它。

从触发器类开始,它包含一个根据以下条件打开和关闭的灯
无论玩家是否在触发器上,我们所做的是首先检测它何时发生碰撞
使用触发器,玩家可以打开门 (因为它在行动范围内
扳机)。

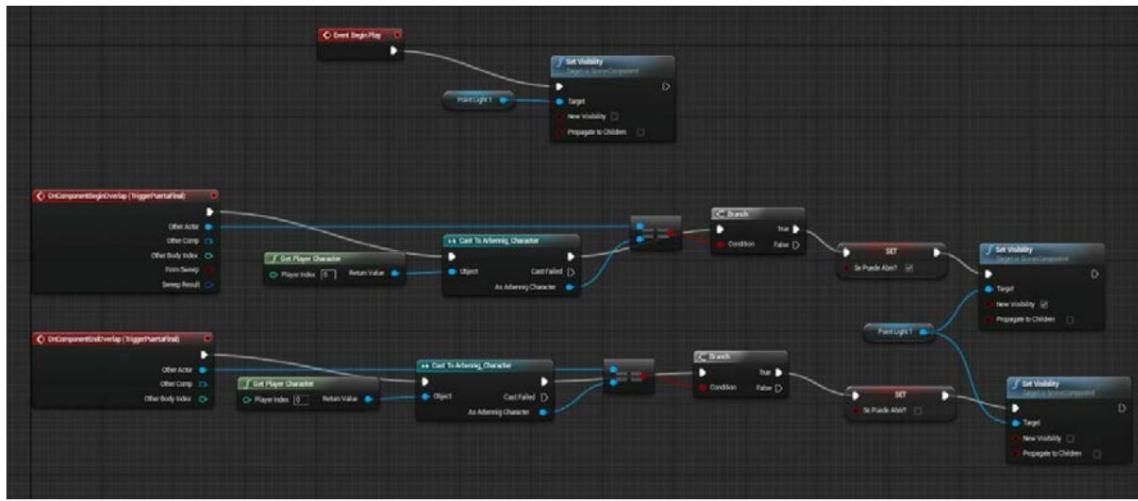


图 167. 捕获角色穿越时的初始化和事件响应

停止通过触发器。

来源:自制。

知道它是否在触发器内,在滴答事件内我们将检查它,以及是否玩家正在尝试与之交互,如果是,我们将检查它是否拥有所有

钥匙,将信号发送到必须打开门的关卡蓝图。

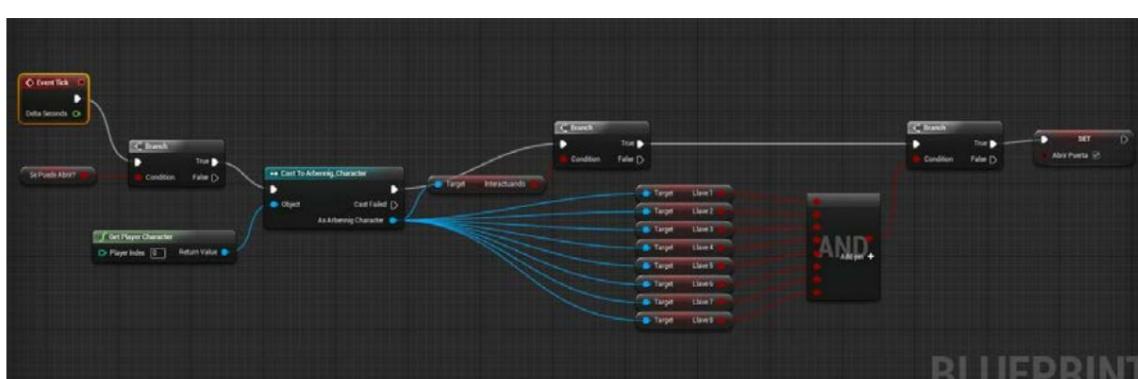


图 168. 在 tick 函数中捕获触发开门的逻辑。

来源:自制。

现在,在 LevelBlueprint 中,我们要做的是让 tick 函数检查是否

我们是否修改了充当标志的变量,以便门打开,如果是,我们将信号发送到门以使用另一个标志打开。

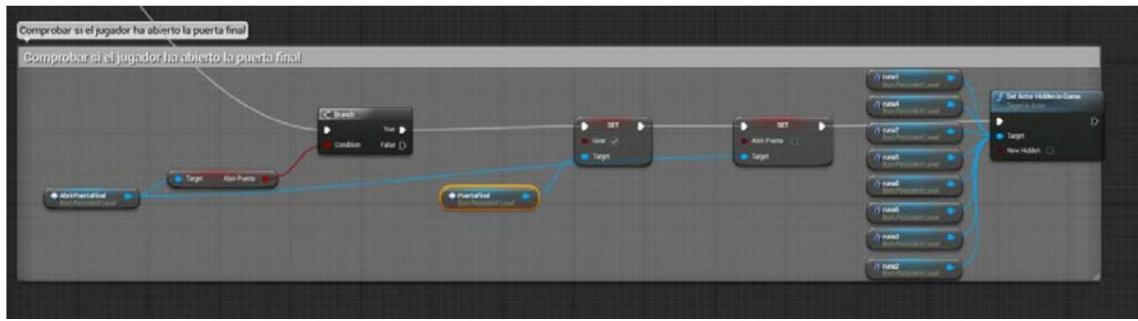


图 169. 关卡蓝图中的开门逻辑捕获。

来源:自制。

最后,在门蓝图中,我们要做的第一件事是它的开始事件

播放,保存这个的初始旋转,我们需要能够使用时间轴。最后,在对 tick 事件的检查中,我们检查其标志是否为

open 已更新为 true,如果是,将使用时间轴,它是一个组件

制作简单动画的蓝图,并依靠插值,我们将旋转门直到它打开。

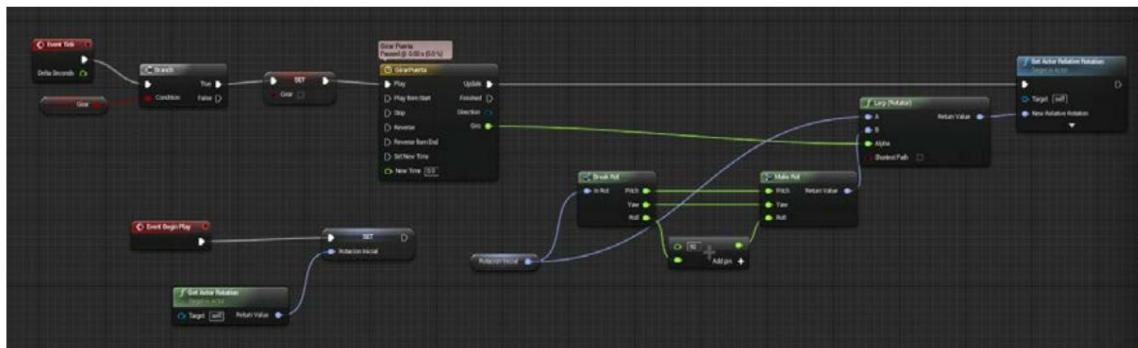


图 170. 最终门类的逻辑捕获。

来源:自制。

作为时间线的额外细节,它们的操作由动画曲线组成

它可以使变量或向量的值根据它而变化。在我们的案例中,

因为我们只需要门打开一个轴,用一个浮点变量作为参数

修改其值的输出对我们来说就足够了。

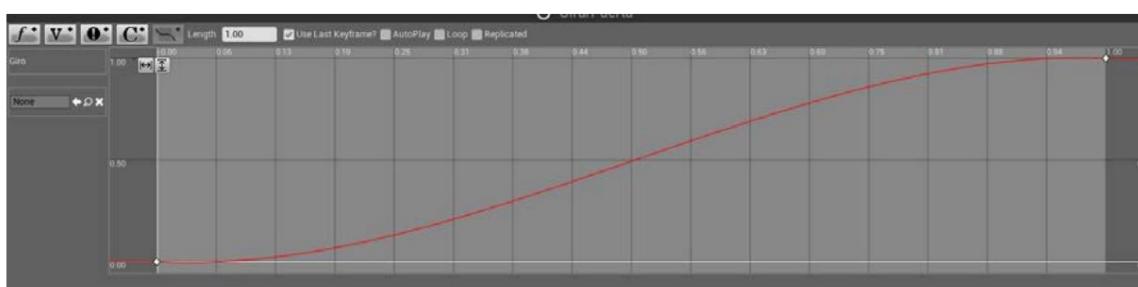


图 171. 捕捉转动门的时间线。

来源:自制。

5.2.7.游戏保存/加载系统。

虚幻引擎 4 中的保存系统是由继承自类的蓝图完成的
保存游戏。

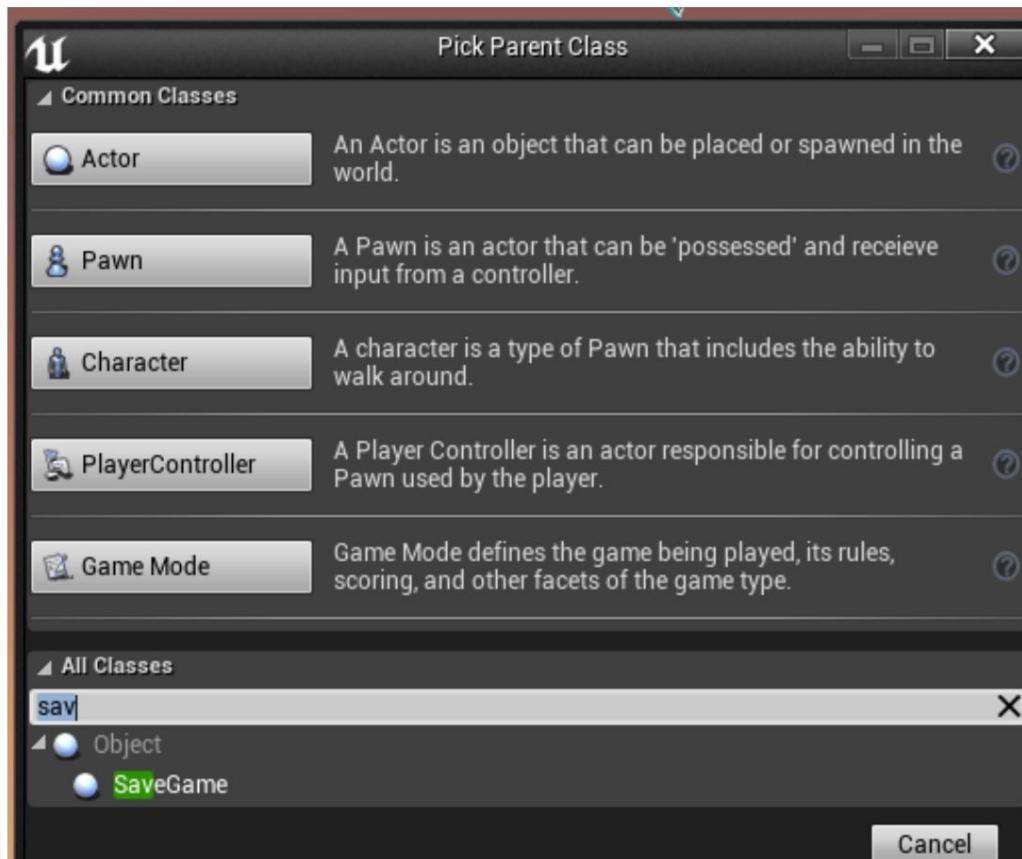


图 172. 如何找到 SaveGame 类的屏幕截图。

来源:自制。

创建这个类后,我们所做的就是在其中存储我们想要保存的所有数据,
为它创建变量,就好像它是一个普通的蓝图一样。在我们的案例中,我们
我们有兴趣保存 8 个布尔值,每个键一个,根据是否为真或假
玩家是否达到了它们,我们还将保存一个向量来存储位置
您保存游戏的玩家,一个布尔值,用于确定是否有游戏
保存,最后,一个字符串变量和另一个我们用来识别文件的整数
保存创建的文件。所有这些变量都必须在其余部分可见
类,所以他们可以访问他们的值。

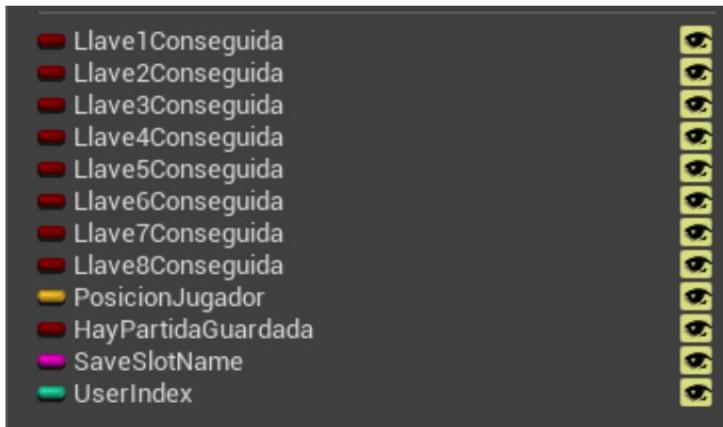


图 173. 捕获存储在蓝图中的变量。

来源:自制。

有了这个类,我们现在可以保存游戏并加载它。

5.2.7.1. 保存游戏。

为了保存游戏,我们创建了一个包含保存点的蓝图,该保存点

包含玩家将出现的位置球。在其中,当玩家靠近时,

将设置一个标志为真,这将在其刻度函数中指示它可以保存游戏。

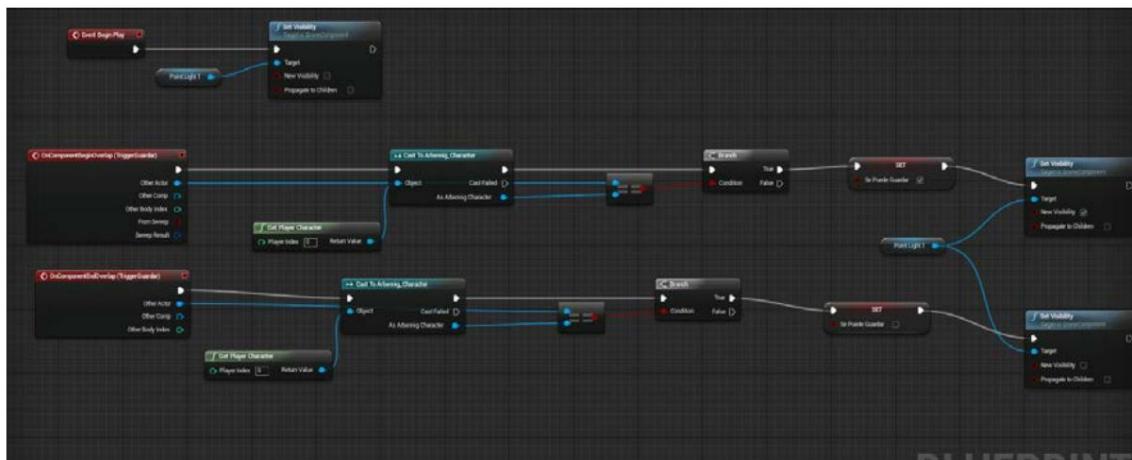


图 174. 检测玩家是否在保存游戏触发器内的逻辑。

资料来源:个人阐述。

完成后,在检查玩家正在交互之后,

保存系统由调用函数创建游戏保存对象组成

我们将指出我们创建的蓝图类,然后,从这个类型的游戏对象知道,

我们对我们的蓝图进行演员表,我们现在将保存演员表的可变结果

访问它包含的属性并修改它们。表示有一个保存的游戏,访问我们的角色,反过来,我们将访问它的布尔键

实现,保存在位置矢量中,再现位置球的位置矢量

我们已经在蓝图中创建了,最后,指示我们要在索引中保存游戏
0,它的名字将是 0。

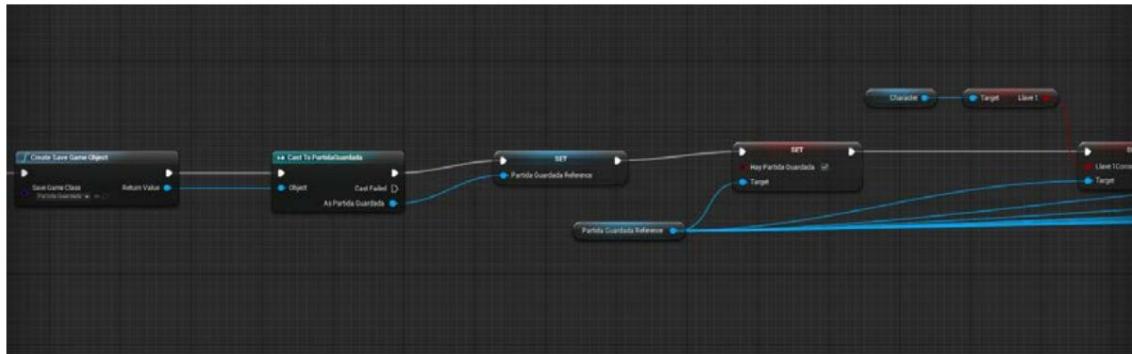


图 175. 保存游戏对象的创建、其转换为保存的游戏以及它的捕获
开始为你的变量赋值。

来源:自制。

当我们完成分配变量后,我们调用函数 save game 到 slot,然后
传递我们保存游戏的新对象,我们已经存储了游戏。

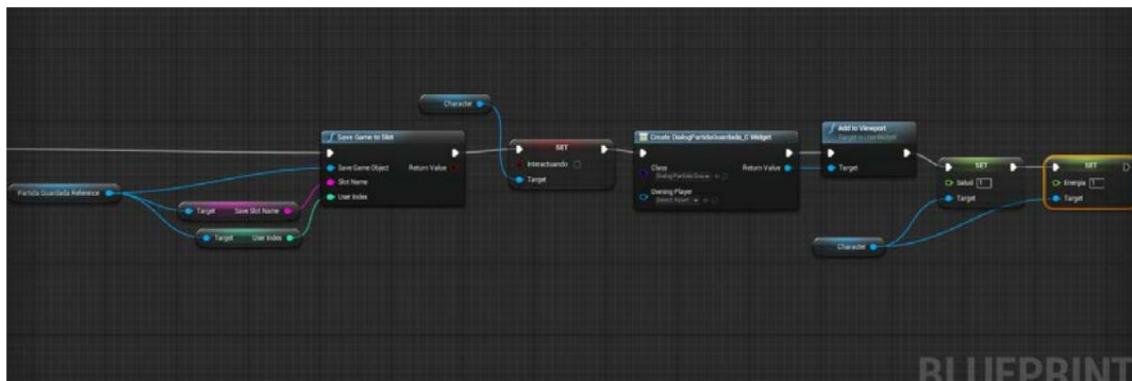


图 176. 对 save game to slot 函数的调用的捕获。

来源:自制。

在此之后,我们在此保存点蓝图中所做的是调用
信息保存游戏小部件蓝图,并恢复玩家的健康和精力充沛
特点。

5.2.7.2 加载游戏。

游戏的加载过程遵循与保存过程非常相似的过程。

我们加载游戏的方法是创建一个保存游戏对象,将其投射到我们的蓝图,然后存储这个投射 (就像在保存中一样)。
完成后,我们调用该函数
从插槽加载游戏,我们将在名称中指示 0,在用户索引中指示 0 (已决定

保存游戏),以及将返回给我们的对象,我们将其转换为保存类的蓝图
游戏,并将其分配给我们已有的参考。

在此之后,我们现在可以访问我们保存的所有变量。

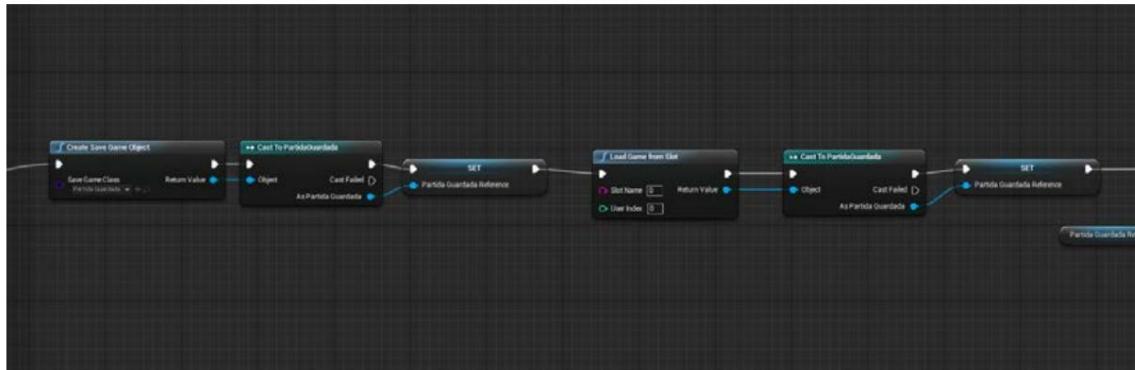


图 177. 加载游戏。

来源:自制。

5.2.7.3。删除游戏。

删除游戏只需要在槽函数中调用删除游戏,说明
插槽和用户索引,在我们的例子中是 0 和 0。

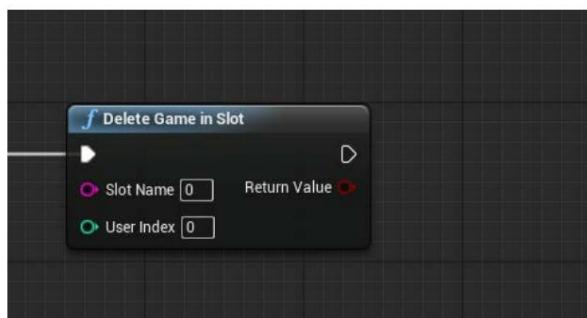


图 178. 删除保存的游戏功能。

来源:自制。

5.2.8。人工智能实施。

GDD 中定义的 AI 的实现需要创建一系列组件
虚幻引擎 4 中的基础知识,我们继续定义和解释它们的设置
基本的,因此稍后我们将专注于区分设计文档中描述的每种人工智能的那些特殊性。

在此引擎中创建 AI 所需的基本组件有 4 个,可以根据 AI 的复杂性增加,它们如下:

1. 行为,它允许我们基于一系列建立行为
条件,并为人工智能做好准备,以允许管理
更有效和图形化的行为。该组件可在部分中找到
德杂项。
2. 黑板,用于存储我们与行为一起使用的变量
树。与行为树一样,该组件也可以在
杂项部分。
3. Character 类型的蓝图。此 Actor 对应于人工智能控制器将拥有的角色,遵循相同的虚幻方案

我们在 Arbannig 中使用,我们的 Player Controller 有 Character Arbannig。

4. 拥有 Character 类型蓝图的 AIController 类型蓝图（可以是
如果它是一个非常特定的 AI,则属于 actor 类型,但不是我们的情况）。

一旦创建了这些组件,我们就开始添加它们的游戏逻辑。在我们的例子中,逻辑
我们将要使用的 AI 游戏,由一个一直保持不动的角色组成
看到玩家,然后追他直到追上。

对于AI角色,我们要做的是添加一个覆盖玩家的触发器。对此
trigger 我们将给它一个功能,如果它与玩家碰撞,那么玩家
将其位置更改为指示的位置,该位置对应于它将出现的丢失区域
比赛。

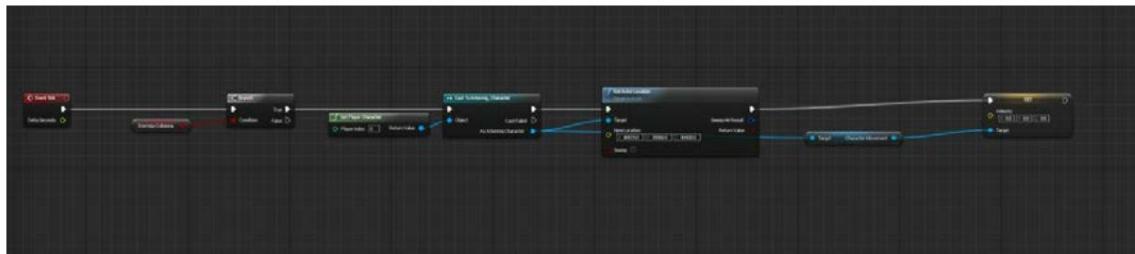


图 179. 捕获敌人碰撞时的位置,我们将其位置更改为指示的位置。

来源:自制。

另一方面,在黑板上,我们创建了 3 个变量,其中一个类型为 object,对应于
要追求的目标,以及两个向量,一个为当前位置,一个为当前位置。
移动到的目标。

当我们创建这些变量时,我们保存并将黑板分配给行为树。

行为树是具有特殊功能的图。在其中,有一个初始节点称为
root,从这个节点往下走其他节点,从左到右执行。在
在此图中,可以创建选择器和序列,它们从左到右执行子节点。

直到一个以成功（选择器）或失败（序列）结束，这导致它上升
那个树。在行为树上执行的条件和检查接收
装饰器的名称（有条件的）和服务（更新黑板并进行检查），
当满足某些特征时我们希望执行的任务，
它们被称为任务。

我们的行为树如下：

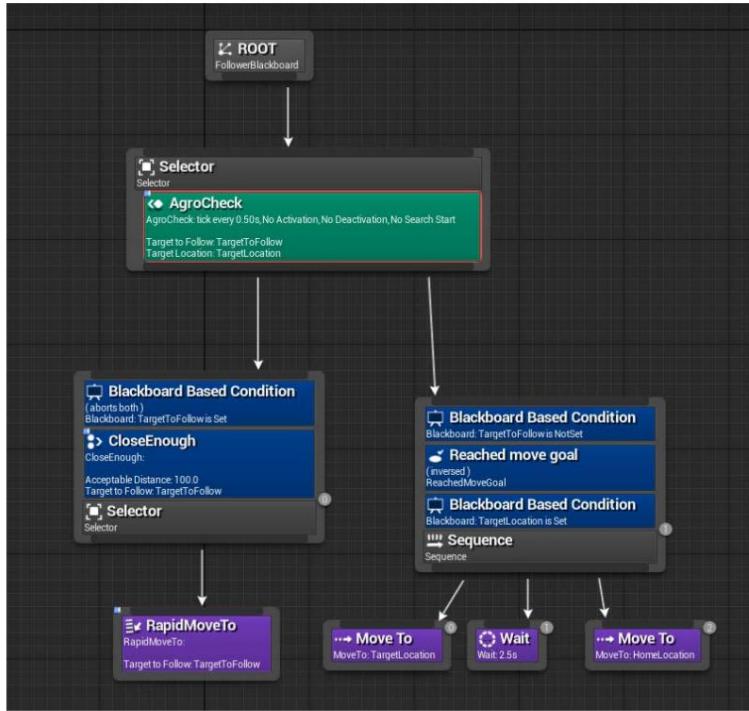


图 180. 捕获我们 AI 的行为树的一般逻辑。

来源：自制。

它在于，从根节点开始，我们有一个选择器，它在每个时刻都会检查 Pawn AI 将要拥有的那个被分配了，如果没有，它就会分配它。此外，它还负责画一系列线来发现一个（可能错过的）要追逐的目标。这个的两块黑板启动，每块黑板检查玩家是否被分配的情况下已分配，并且它足够接近（装饰器 CloseEnough 检查距离），然后启动追逐任务。另一方面，如果玩家不已分配，尚未到达其目的地，但有一个要移动到的目的地（在看不见我们的那个），它所做的就是回到它计划去的位置，等待 2.5 秒，然后回到它原来的位置。

为了让所有这些工作，在编辑器中，我们要做的是创建一个导航网格绑定元素，这将是角色将通过的导航网格。

5.2.9.在关卡蓝图中实现游戏的一般游戏流程

关卡蓝图中的一般游戏流程分为几个部分。

首先,在创建关卡时,调用 begin play 事件,它执行以下操作:

1. 让 PlayerController 拥有 Pawn 并将其存储在一个变量中。

2. 加载游戏。

3. 为我们有多个对象的情况创建一个元素数组

相同的类 (对于篮球和 AI) ,并隐藏舞台的所有元素

应该是 (来自挑战 4 和 7 的篮球,来自挑战 2 的传单,来自挑战 6 的傀儡,
挑战 8 面旗帜,以及符文墙上的符文) 。

完成此操作后,tick 函数将检查我们是否处于挑战中。为了这
我们创建了一个有 9 个值的枚举器,其中 8 个表示挑战,还有一个,即
处理 none 值,因为当它是 none 时。由于我们要学习的字符类
用作从游戏的任何蓝图轻松访问的桥梁,我们创建一个变量
无论是在关卡蓝图中还是在这种类型的枚举器的性格中。

在每一时刻,我们遵循的逻辑如下:

1. 检查枚举数的值,如果不等于none,则表示有
一个挑战。

2. 检查字符枚举变量是否已经更新为挑战
当前,如果没有,则将值分配给我们的关卡蓝图。

3. 初始化相应的挑战。
一个。对于挑战 1、3 和 5,您无需初始化任何内容。

湾。对于挑战 2,我们将在镇上分发的小册子是可见的。

C。对于挑战 4 和 7,我们创建了箍计时器小部件,它

将其存储在变量中,并将其添加到视口,然后,像这样

小部件我们将实现的箍初始化为0,我们将目标箍分配给

通过传递我们在开始播放中创建的环数组的长度来实现,

我们指明了克服挑战的时间,最后指明了目的地

挑战结束时传送。

d。对于挑战 6,我们让玩家必须获得的旗帜和

魔像。

和。对于挑战 8,玩家必须获得的旗帜是可见的。

4.检查是否满足完成每个挑战的条件,以及

如果是这样,请调用负责生成符文的自定义事件

对应已通过的挑战,并显示 Challenge Blueprint Widget

克服。

一个。挑战 1 和 5 直接调用结束挑战事件,因为

当您完成与图腾的对话时,挑战就开始了。

湾。挑战 2 呼吁在所有讲义完成后结束挑战

发表。为了确定这一点,我们有一个在宣传册交付时设置为 true 的标志,以及一个引用触发器的变量。

玩家在哪里,当传单送达时,我们将它添加到一个数组中

关卡蓝图中的布尔值,我们销毁角色所在的触发器,使其无法再次向其传递小册子,当长度

数组的数量达到传单的数量,则意味着所有

小册子已分发。

C。挑战 3 会在所有灌木都放置在指定位置时这样做

在任务中。在挑战 3 结束时,因为他们所在的触发器已经

包含所有这些,在字符中创建的变量设置为 true,这

我们阅读了关卡蓝图。

d。挑战 4 和 7 在完成后调用结束挑战事件

在指定时间内所有的箍。这些挑战的逻辑在于

确定何时遍历循环,这会将标志设置为真,所以

这样在每个时刻,环数组都会被遍历,并添加到一个变量中

遍历的每个循环标志的整数 +1,并且在遍历完成时更新

计时器小部件变量和剩余的环。当小部件

通知我们挑战已经结束,如果成功,如何调用

结束挑战事件,而如果它没有克服它就结束了,一切都会被重置,

将图腾挑战枚举器设置为无,设置为 false

箍的所有变量都交叉,并在游戏中隐藏箍。

和。对于挑战 6,当在

在放置角色告诉我们的位置后,我们的角色设置为真

图腾要收集的旗帜。另外,剧中的演员

如果已经克服了挑战。另一方面,如果没有挑战

超出(用字符中的另一个变量表示),我们做什么

是重新开始挑战,将旗帜重新定位并隐藏在原来的位置,

傀儡也是一样。除了还将枚举数更改为

无状态,并显示挑战尚未通过的小部件。

5. 检查玩家是否死亡,如果是,打开残局关卡并移除

平视显示器。

6. 检查是否生成了符文,如果是,请检查玩家是否生成了它

通过检查它是否是一个有效的对象已经被拾取。根据所面临的挑战

我们正在做,我们摧毁了允许我们与说的图腾交谈的触发器

挑战,然后,我们将活动挑战枚举器设置为无。

7. 最后,我们检查玩家是否打开了最后一扇门,即,

检查我们在触发器中放置符文的布尔值是否已设置为 true,如果为 true,我们调用门旋转并使粘合的符文可见

到最后一堵墙。

6. 结论

完成项目后,我们可以突出一系列关于它的结论和
此外,对使用几个月后使用的工具进行评估。

应该指出的是,可能或多或少地符合假设的目标
在报告的第 3 部分,即视频游戏的完整设计和
创建后,它已使用虚幻引擎 4 引擎实现,涵盖了各个方面
游戏世界的创建及其完整的逻辑,它还添加了游戏菜单,
序幕和尾声的过场动画,已与外围眼裂隙 DK1 兼容,并且是
已经敲响了这个项目。有了这一切,我们可以确定得到的结果是
正确的。

除此之外,经过几个月使用蓝图脚本系统的开发,
可以肯定它是一个非常强大的系统,它允许我们制作视频游戏
100% 完成,无需接触任何代码。虽然是的,蓝图系统
它是图形脚本,也就是说,我们在代码中拥有的蓝图中用
节点以更直观的方式拥有一切。有了这个,我想强调一个人
没有编程知识或只有基本知识,您可能会
开发项目非常复杂,因此设计师的个人资料或
想要使用它的艺术家,必须具备面向编程的知识
对象和程序遵循的执行逻辑。

另一方面,引擎的学习曲线有点棘手
一些方面取得进展。尽管使用 UDK 引擎确实有所改进
毫无疑问,像 Unity 这样的引擎提供的学习曲线相对陡峭。
简单,尤其是开始使用它。

就引擎提供的可能性而言,很明显为什么它是其中之一
世界上最相关和最著名的引擎,以及它包含的所有工具,界面
比在 UDK 中更容易理解,而且强大的视觉效果
让我们的表演非常了不起。

值得一提的是,尽管他之前曾与其他引擎进行过项目
视频游戏,包括 Unity 和 UDK,我之前从未使用 UE4 做过项目,
并且所有的机制和视觉部分,以及一般的一切,都是从头开始学习的
马达。

作为一个否定的结论,我必须指出,在开发过程中,引擎一直遭受强制关闭,有时在我专注的日子里达到每天 10 次左右的水平项目完全。除此之外,通常在项目关闭时出乎意料,已经无法恢复进度了,所以很方便保存对项目所做的每几个更改。

除此之外,已经注意到虽然多媒体工程师的能力让你成功完成一个电子游戏,你所拥有的艺术技能是相对有限,除非你对它们有一定的实践和热爱,所以在开发更专业剪辑的视频游戏时,如果你想能够开发项目,那么进行一些辅助艺术培训会很方便。一个人。

6.1.改进建议和未来工作。

作为改进和未来工作的建议,除其他方面外,主要包括:
以第一人称执行角色的建模和动画,其网格和动画是引擎默认为 FPS 项目提供的。

另一方面,让玩家可以分别控制音乐和游戏声音的音量,将项目中使用的免费虚幻资产替换为拥有使用 3ds max 等建模程序创建的资产,改进控制游戏和添加 Xbox 控制器支持也将是改进的建议。
此外,对于未来的工作,我们将努力提高他们步履蹒跚的技能,这在艺术部分,或与有艺术技能的人一起表演专业人士。

7. 参考书目和参考文献

7.1. 介绍。

[1] 电子游戏的历史：

https://es.wikipedia.org/wiki/Historia_de_los_videojuegos#Balance.2C_presente_y_futuro_de_los_videogames

[2] 电子游戏计费：

http://elpais.com/diario/2008/04/09/radiotv/1207692005_850215.html

[3] 命运史上最昂贵的电子游戏：

<http://www.abc.es/tecnologia/videojuegos/20140909/abci-destiny-gameplay-videojuego-RPG-fPS-Activision-201409091206.html>

[4] Flappy Bird 成功：

https://es.wikipedia.org/wiki/Flappy_Bird

[5] 阿利坎特大学能力多媒体工程师：

<http://cvnet.cpd.ua.es/webcvnet/planestudio/planetudiond.aspx?plan=C205>

7.2. 理论框架或最先进的技术。

[6] Bernard Suits 的电子游戏定义：

蚱蜢：游戏、生活和乌托邦，伯纳德套装，1978。

[7] 维基百科的视频游戏定义：

<https://es.wikipedia.org/wiki/Videojuego>

[8] 发动机的历史

游戏引擎架构，Jason Gregory，2015 年（第二版）

[9] 虚幻引擎创建于 1998 年：

https://es.wikipedia.org/wiki/Unreal_Engine

[10] Source 引擎于 2004 年推出：

<https://es.wikipedia.org/wiki/Source>

[11] 游戏引擎列表：

https://en.wikipedia.org/wiki/List_of_game_engines

[12] 标志虚幻引擎：

https://upload.wikimedia.org/wikipedia/commons/e/ee/Unreal_Engine_logo_and_wordmark.png

[13] 标志统一：

https://upload.wikimedia.org/wikipedia/commons/5/55/Unity3D_Logo.jpg

[14] 徽标 CryEngine：

[https://upload.wikimedia.org/wikipedia/commons/8/8d/CryEngine_Next_Gen\(4th_Generation\).png](https://upload.wikimedia.org/wikipedia/commons/8/8d/CryEngine_Next_Gen(4th_Generation).png)

[15] UE4 vs Unity vs CryEngine：

<http://blog.digitaltutors.com/unity-udk-cryengine-game-engine-choose/>

[16] 主题“视频游戏引擎”幻灯片中关于引擎选择的建议

电子游戏 2。

https://moodle2014-15.ua.es/moodle/pluginfile.php/18327/mod_resource/content/4/vii-08-引擎.pdf

[17] 基本引擎架构

<https://docs.unrealengine.com/latest/INT/Programming/UnrealArchitecture/index.html>

[18] 电机类图的图像

<https://docs.unrealengine.com/latest/images/Gameplay/Framework/QuickReference/GameFramework.jpg>

[19] 蓝图信息

<https://docs.unrealengine.com/latest/INT/Engine/Blueprints/index.html>

<https://docs.unrealengine.com/latest/INT/Engine/Blueprints/Overview/index.html>

7.3.方法[20] 什么是
看板以及如何使用：

<http://www.javiergarzas.com/2011/11/kanban.html>

7.4.视频游戏设计文档。

[21] Fidel Aznar 的电子游戏基础知识 GDD 模板。

[22] PEGI 标志：

https://upload.wikimedia.org/wikipedia/commons/thumb/4/44/PEGI_12.svg/426px-PEGI_12.svg.png

[23] PEGI12信息：

<http://www.pegi.info/es/index/id/96/>

[24] 维基百科的游戏玩法定义：

<https://es.wikipedia.org/wiki/Jugabilidad>

[25] 最低软件和硬件规格在网络上的常见问题解答中描述

UE4：

<https://www.unrealengine.com/faq>

7.5.开发和实施。

[26] UE4中的静态网格导入管道：

<https://docs.unrealengine.com/latest/INT/Engine/Content/FBX/StaticMeshes/index.html>

7.5.1.经常查阅的参考书目

虚幻引擎文档网站：

<https://docs.unrealengine.com/latest/INT/index.html>

官方虚幻引擎 4 youtube 频道：

<https://www.youtube.com/channel/UCBobmJyzsJ6Ll7Ubfhl4iwQ>

虚幻引擎 4 论坛：

<https://forums.unrealengine.com/>

UE4 答案中心：

<https://answers.unrealengine.com/>

UE4 音频规格：

<https://docs.unrealengine.com/latest/INT/Engine/Audio/WAV/index.html>