

ДОПЪЛНИТЕЛНИ РЕСУРСИ, ПОДПОМАГАЩИ АВТОМАТИЧНОТО СЪЗДАВАНЕ НА ОБКРЪЖЕНИЕТО В 3D ИГРОВИЯ ДИЗАЙН В ПОМОЩ НА ОБУЧЕНИЕТО

Ивайло Ив. Буров

Шуменски университет „Епископ Константин Преславски“

***Резюме:** Тази статия доразвива концепцията за създаване на 3D екстериора, разгледана в монографичния труд „Изграждане на дигитална среда за интерактивно 3D игрово обучение. Разработка и интеграция на интерактивни 3D решения“ - където предимно са засегнати ресурси, генериращи обкръжение извън игровия двигател, и статията „Базови ресурси, подпомагащи автоматичното създаване на обкръжението“, където са представени базови ресурси, генериращи обкръжение директно в игровия двигател Godot.*

Представените материали могат да бъдат полезни за създатели на 3D игрово съдържание за целите на обучението.

***Ключови думи:** 3D ресурси, 3D обкръжение, екстериор, игрови дизайн, автоматично генериране, Godot*

ADDITIONAL RESOURCES FOR AUTOMATED ENVIRONMENT GENERATION IN 3D GAME DESIGN TO SUPPORT TRAINING

Ivaylo Iv. Burov

Shumen University "Ep.K.Preslavski"

***Abstract:** This article further develops the concept of creating a 3D exterior, discussed in the monographic work „Building a digital environment for interactive 3D game training. Development and integration of interactive 3D solutions“ – where resources generating environments outside the game engine are mainly affected, and the article „Basic resources supporting automatic environment creation“, where basic resources generating environments directly in the Godot game engine are presented.*

The presented materials may be useful for creators of 3D game content for training purposes.

***Keywords:** 3D resources, 3D environment, exterior, game design, automatic generation, Godot.*

В спектъра на съвременния игрови дизайн попадат множество задачи, за решаването на които са необходими групи специалисти, специализирани в отделни направления, включващи създаване на графика и ефекти, моделиране и анимация, създаване на затворени и отворени светове, игрови персонажи, сценарии, сюжети и др.

Поради множеството високи изисквания за създаването на игрово съдържание с добро качество крайният продукт е свързан с високи разходи откъм труд, ресурси и финанси. Ако насочеността е към т.нар. сериозни игри, към които спадат и учебните, са необходими и допълнителни методически насоки от специалисти в съответното образователно направление и привличането им към работещия екип. Финансирането на такива проекти обикновено е свързано с много по-големи затруднения, отколкото при забавните видеоигри, при които комерсиалните компании са по-склонни да инвестират средства. Още повече при създаването на интерактивно игрово образователно съдържание е необходимо разработваните сюжети и сценарии да бъдат синхронизирани както с учебното съдържание, така и с възрастовата група на обучаваните. И. Бурова предлага разработването на такива сценарии чрез алгоритмично представяне. „**Под алгоритмично представяне на сценария** се разбира сюжетно описание на последователността от събития и техните причинно-следствени връзки с добавено съкратено описание на диалозите от сценария, определящи хода на играта“ (Бурова, 2022, стр. 77). Авторът посочва необходимостта от баланс между учебната и забавната част в сценария. Докато за методическата част в образователната система са подготвени множество специалисти, в създаването на интерактивно образователно игрово съдържание е по-трудно да се намерят такива, защото болшинството от тях вече са привлечени в създаването на комерсиално забавно игрово съдържание, където финансирането е по-голямо. Поради тази причина и други – като необходимостта от пресечна област между техническата и образователната част, възниква необходимост от инструменти за автоматично създаване на елементи от игровото обкръжение с цел намаляване на разходи и усилия при разработката му.

В монографичния труд „Изграждане на дигитална среда за интерактивно 3D игрово обучение. Разработка и интеграция на интерактивни 3D решения“ (Буров, 2022) предимно са засегнати решения, генериращи обкръжение извън игровия двигател, на база специализиран или стандартен софтуер за създаване на модели за обкръжението, а в статията „Базови ресурси, подпомагащи автоматичното създаване на обкръжението в 3D игровия дизайн в помощ на обучението“ (Буров, 2024) са изследвани ресурси за създаване на 3D игрово обкръжение, достъпни за работа директно в игровия двигател Godot, като към базовите ресурси са причислени решения за процедурно генериране на терени, а в настоящата статия са представени допълнителни ресурси, отнасящи се до генерирането на пътища, растителност, водни повърхности, сгради, пещери и подземия. Разгледани са решения, съвместими с игровия двигател Godot, изборът на който е обоснован в статията „Подбор на игрови двигатели при разработка на 3D интерактивно съдържание в обучението“ (Буров, 2021).

Godot Road Generator е добавка за лесно създаване на 3D магистрали, улици и пътища. Плъгинът е наличен за версии на Godot 4.2 – 4.3, както и за версия 3.5. Добавката е с отворен код под MIT лиценз и е налична както в github хранилището, така и в хранилището за активи на Godot.

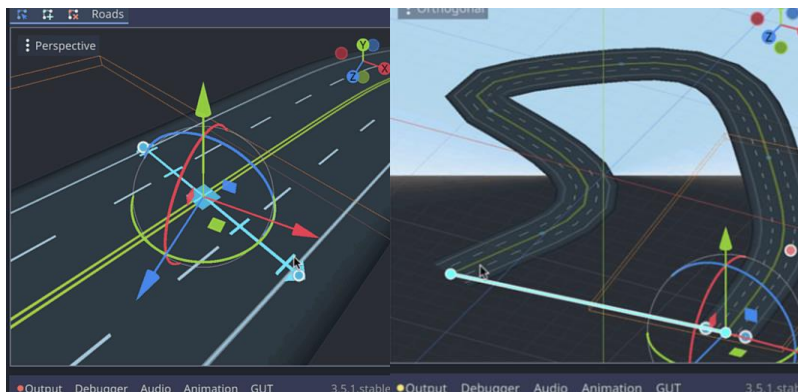
Авторите на приставката описват следните възможности:

- пълноценно персонализирана геометрия, разширяваща 3D Path възела, за перфектно съгласувани шевове на пътя без дупки или пропуски (Фиг. 1);
- персонализирани възли за генериране на пътища (RoadManager, RoadContainer, RoadPoint и RoadLane);
- помощен възел RoadLaneAgent за лесно следване на пътя;
- възможност за дефиниране на произволен брой ленти и други параметри на базата на точка от пътя (RoadPoint). (Фиг. 2);

- процедурна геометрия за смяна на ленти въз основа на свързани пътни точки (RoadPoints) (Фиг. 2);
- автоматично генериране на колизионни мрежи;
- автоматично създаване на AI маршрути за ленти с връзки между съседни елементи;
- визуална обратна връзка с персонализирани приспособления в прозореца за изглед;
- бързо генериране на пътища с интерактивен режим за добавяне и инструмент за свързване на живо;
- разнообразие от предварително моделирани кръстовища, които могат да се поставят в сцената и да се „закачат“ към други пътни точки.

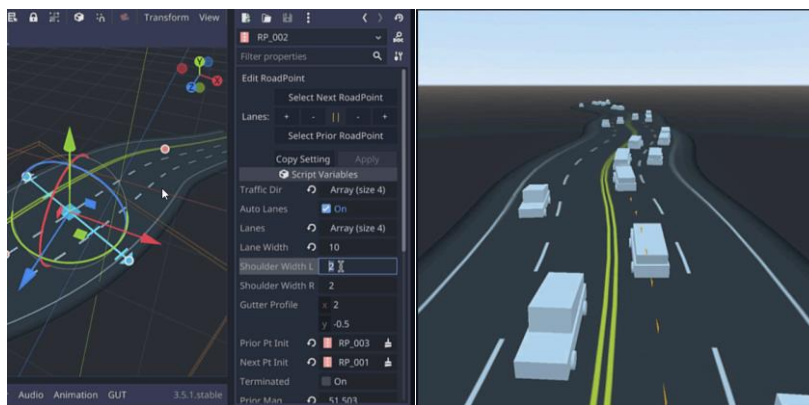
В разработка:

Поддръжка за процедурни кръстовища (вече са налични ръчно моделирани 3D кръстовища).



Фиг. 1. Пътни точки и рисуване на пълни пространствени контроли за фина настройка на разположението

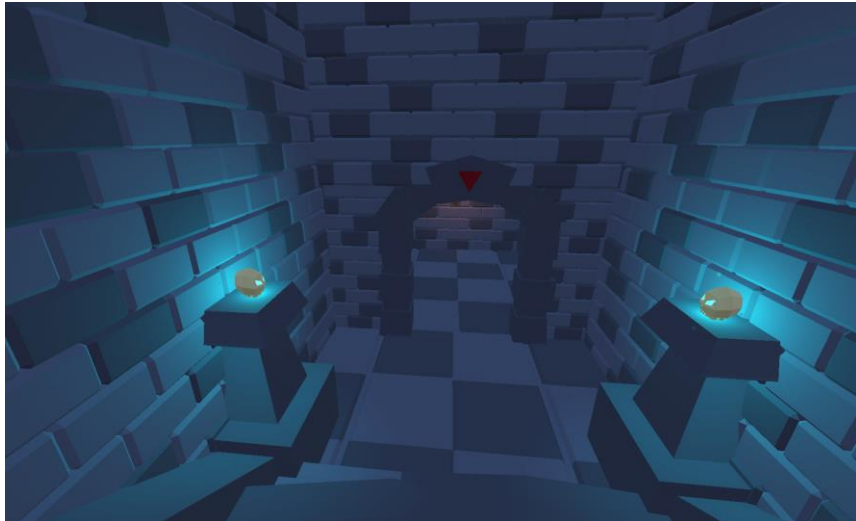
Източник: <https://github.com/TheDuckCow/godot-road-generator>



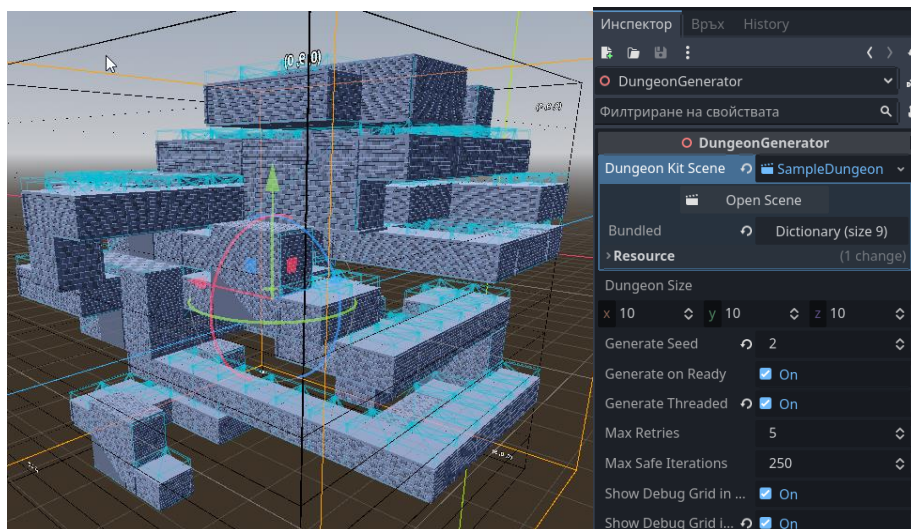
Фиг. 2. Инспекторски панел на пътна точка за определяне на ширина на лентата; Налични по време на изпълнение функции за процедурно използване

Източник: <https://github.com/TheDuckCow/godot-road-generator>

SimpleDungeons (Фиг. 3) е Godot проект с отворен код за генерация на подземия. В официалния проект липсват описания на алгоритмите, по които се осъществява генерацията, като по-подробни данни относно това са представени в Youtube видеа – Procedural 3D Dungeons In Godot 4 Tutorial и Procedurally Generated 3D Dungeons. За разлика от повечето генератори на подземия, при които генерацията се реализира на едно ниво/плоскост, при това решение няма ограничение за броя на нивата в 3D пространството, което позволява създаване на сложен комплекс от лабиринти, наподобяващи сграда с множество етажи (Фиг. 4).

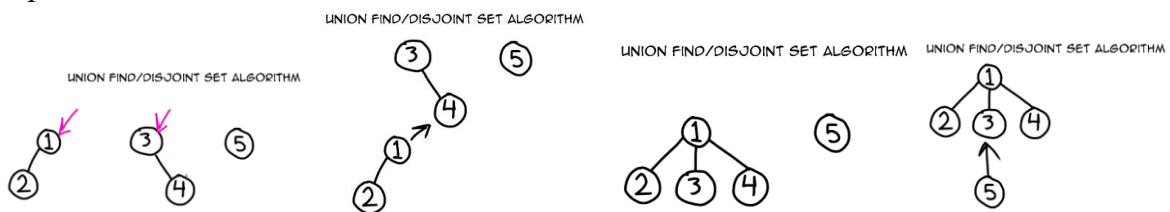


Фиг. 3. SimpleDungeons демо



Фиг. 4. Генерация на подземия с много нива; Инспектор за генерация

Авторът използва алгоритъм за свързване на отделните стаи и етажи, базиран на йерархия от възли, които изграждат дървета, като за корен на дървото използва най-ниската стойност на възела в дървото (Фиг. 5) и по този начин поддържа списък на свързани стаи и етажи.



Фиг. 5. Използване на възли, организирани в дървета за свързване на обектите в сцената
Източник: <https://www.youtube.com/watch?v=TPvxWIKHE6Q>

Подземието се изгражда посредством използване на сглобяеми стаи с предварително дефинирани форми и врати. Първоначално се използват подобни на примитиви сглобяеми елементи за стаи, коридори и стъпала, които могат да се заменят с действителни 3D модели, като се дава възможност за замяната им с конфигурируеми от

потребителя сглобяеми 3D модели. За правилната генерация при използване на потребителски 3D активи е необходимо да се спазват определени условия, описани в Wiki страницата на проекта:

– При създаването на 3D стая за подземие се вижда вокселна мрежа и може да бъдат променени размерите на стаята във воксели. Всички стаи трябва да наследят `DungeonRoom3D`.

– За да се добавят врати към потребителски дефинираната стая, така че да може да бъде свързана с останалата част от подземие, е необходимо добавянето на възли с имена с префикс `DOOR` за задължително или `DOOR?` – опционално.

– При създаване на коридор той трябва да е `1x1x1` във воксели и да има 4 незадължителни врати – по една от всяка страна. Коридорът ще се използва при намиране на пътека за свързване на всички стаи заедно.

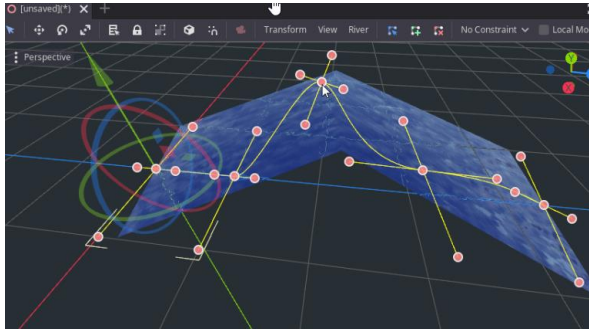
– Вокселната скала на генератора за подземия `DungeonGenerator3D` трябва да съвпада с вокселната скала на всеки от стаите тип `DungeonRoom3D`. Ако мащабът на вокселите не е същият, възлите от тип `Node3D` на стаята ще бъдат мащабирани, навярно неравномерно, така че мащабът на вокселите да съвпада. В конзолата ще бъде отпечатано предупреждение, ако вокселните скали не съвпадат.

Waterways (Фиг. 6) е Godot добавка – инструмент за генериране на речни мрежи с карти на потока и пяна, базирани на криви на Безие. Създадена е от Kasper Arnklit Frandsen с отворен код под MIT лиценз (Frandsen K. Waterways).

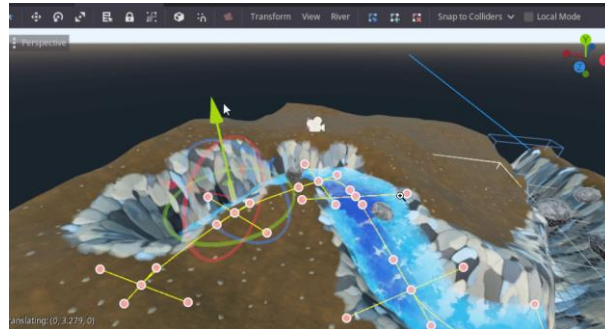


Фиг. 6. *Waterways* демо
Източник: <https://github.com/Arnklit/Waterways>

Приставката съдържа основен възел за създаване на река от тип `River` (Фиг. 7), като след добавянето му към сцената реката може да се моделира посредством `Path` контроли (контроли на пътя) според нуждите на потребителя. Предвидени са ограничители посредством колидери за прилепване на реката към терен (Фиг. 8).



Фиг. 7. Използване на възли *mix River*
Източник: <https://github.com/Arnklit/Waterways>

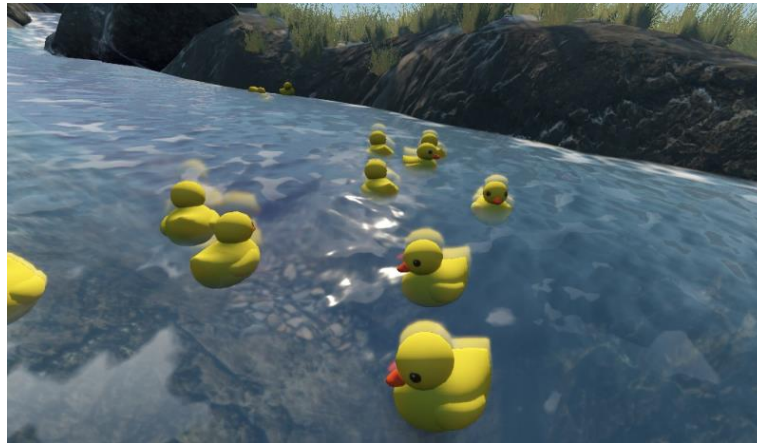


Фиг. 8. Колидери за прилепване към терен
Източник: <https://github.com/Arnklit/Waterways>

След като е завършена формата на реката, може да се използва опцията *River* → *Generate Flow & Foam Map* за „изпичане“ на текстурите, които активират картата на потока и пяната. Настройките за печене се намират в инспектора на реката. За получаване на директен достъп до мрежата на реката, в случай че се използва за други цели като например генериране на форма на сблъсък, може да бъде използвана опцията *River* → *Generate MeshInstance Sibling*.

За генериране на глобална височина и карта на потока на реката може да се добави възел *WaterSystem* и да се генерира текстура на базата на всички дъщерни възли на възела *River* с опцията *WaterSystem* → *Generate System Maps option*.

В приставката е добавен и възел от тип *Buoyant Node* (плаващ възел), добавянето на който като дъщерен към *RigidBody* ще позволи на обект да плава по реката, ако е налична (*WaterSystem*) с валидни карти (Фиг. 9).

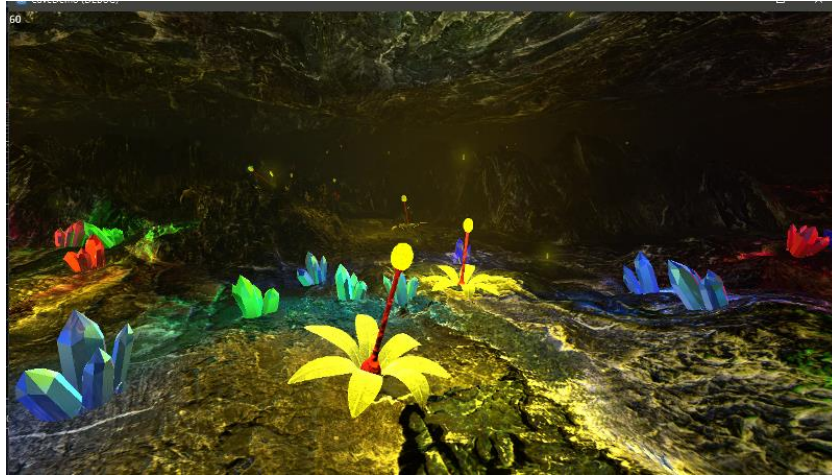


Фиг. 9. Използване на *Buoyant Node*
Източник: <https://github.com/Arnklit/Waterways>

Забележка: Последната актуализация на добавката *Waterways* е от 27.01.2021 г. и съществуват проблеми на съвместимостта с последните версии на *Godot 4.x*, които все още не са отстранени.

Godot Cave demo (IcterusGames) е демонстрационен проект за процедурно генериране на пещери. Проектът не е достъпен в хранилището за активи на *Godot*, а достъпът до него е само чрез *github* страницата на проекта. На официалната страница липсва всякакво описание относно проекта, освен че проектът е с отворен код под *MIT* лиценз. При тестиране на проекта (Фиг. 10) бе установено, че е реализирана безкрайна повтораемост на генерираните пещери. Това е удобно при изграждане на безкраен игрови свят, но при необходимост за допълнително моделиране на сцените и разполагане

на специфични 3D активи в тях често е необходимо изключване на тази опция и превръщането на сцената в редактируема мрежа. Първоначално това не може да бъде осъществено, защото проектът генерира пещерите само по време на изпълнение на програмата, а в режим на дизайн няма никакви достъпни опции освен черен екран.



Фиг. 10. Cave demo по време на изпълнение на програмата
Източник: https://github.com/IcterusGames/cave_demo

С малки промени в кода става възможно генерацията да бъде осъществена в редактора по време на дизайна. Основните стъпки за постигането на това са:

- добавяне на **@tool** в началото main.gd, terrain_chunc.gd, crystal_01.gd за изпълнение в редактора на Godot.
- заместване на реда във функцията:

```
func _on_timer_rebuild_timeout():  
    #node.set_owner(get_tree().node_terrain)    c  
    node.set_owner(get_tree().edited_scene_root) #този ред заменя предния
```

- добавяне ред във функцията:

```
func _build_meshes_finished(x : int, z : int):  
    node_floor.set_owner(get_tree().edited_scene_root) #добаен на ред
```

- добавяне на ред във функцията:

```
func _build_meshes_finished(x : int, z : int):  
    node_floor.set_owner(get_tree().edited_scene_root) #добавен на ред
```

- замяна на ред във функцията:

```
func _build_meshes_finished(x : int, z : int):  
    # node_floor.add_child(plant) c  
    plant.set_owner(edited_scene_root) #този ред заменя предния
```

и добавяне на следните редове

```
if randi_range(0, 100) == 5:  
    var plant : Node3D = PLANT_01.instantiate()  
    node_floor.add_child(plant)  
    plant.set_owner(get_tree().get_edited_scene_root()) #добавен ред  
if CREATE_CEIL:  
    add_child(node_ceil)  
    node_ceil.set_owner(edited_scene_root) #добавен ред
```

- добавяне на ред във функцията:

```
func create_floor(world_x : int, world_z : int)  
if CREATE_PHYSICS:  
    # Fisicas por medio de HeightMapShape3D  
    .....
```

```
shape.set_owner(get_tree().get_edited_scene_root()) #добавен ред
```

- добавяне на ред във функцията:

```
func create_ceil(world_x : int, world_z : int):  
if CREATE_PHYSICS:
```

```
.....  
static_body_ceil.add_child(shape)
```

```
shape.set_owner(get_tree().get_edited_scene_root()) #добавен ред
```

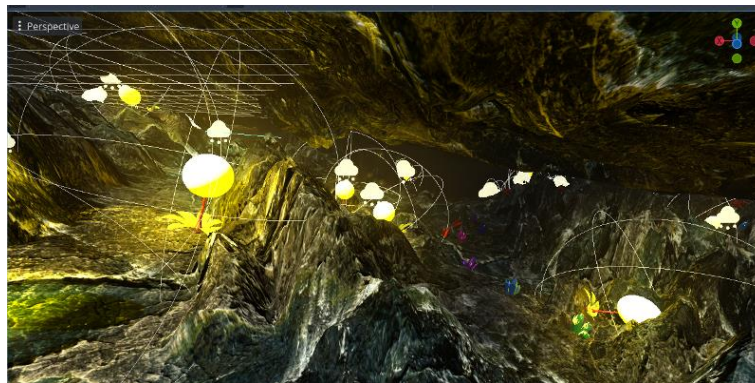
- при проблеми с цвета на генерираните кристали може да се добавят следните редове във функцията:

```
func _ready():
```

```
$Cristal01.get_active_material(0).albedo_color = color #добавен ред
```

```
$Cristal01.get_active_material(0).emission = color #добавен ред
```

След извършените модификации в кода генерацията се извършва в редактора на игровия двигател (Фиг. 11) и генерираното съдържание може да се експортира в glTF файлов формат, достъпен за модификация с външен софтуер за триизмерно моделиране.



Фиг. 11. Cave demo в редактора на Godot след модификациите на кода

Кодът процедурно генерира пещера, която се състои от разделна горна и долна част. Такова решение е удобно при съставяне на миникарта за движението на игровия персонаж по време на изпълнение на играта. В допълнение към генерацията са включени реализации на автоматично разположение на кристали, цветя, светлини и други обекти, които могат да бъдат поместени на сцената. При разширяване на кода, отговарящ за тяхното разположение, могат да бъдат добавени и допълнителни елементи от планирания дизайн.

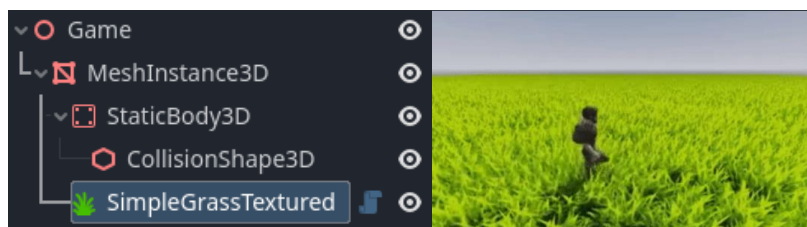
SimpleGrassTextured (IcterusGames) е приставка към игровия двигател Godot, която позволява добавянето на трева и растения към сцената по лесен начин (Фиг. 12). Приставката съдържа възел SimpleGrassTextured, който трябва да бъде поставен като дъщерен на възела за терен (Фиг 13). Това позволява рисуването на трева върху терена.



Фиг. 12. SimpleGrassTextured demo

Източник: <https://github.com/IcterusGames/SimpleGrassTextured>

В допълнение поддържа интерактивен режим, при който тревата може да взаимодейства с движещи се персонажи и обекти в сцената (Фиг. 13).



Фиг. 13. Възел SimpleGrassTextured и интерактивен режим

Източник: <https://github.com/IcterusGames/SimpleGrassTextured>

За целта във функцията `_ready` на игровата сцена трябва бъде активиран интерактивния режим чрез извикване на функцията `SimpleGrass.set_interactive(true)`.

След това трябва да се разреши откриването на движещия се персонаж от системата за колизия на приставката `SimpleGrassTexture`. Препоръчва се добавянето на `MeshInstance3D` инстанция към персонажа – примерно сфера с височина като на тревата.

В слоевете за рендиране на `MeshInstance3D` трябва да бъде активиран само слой 17. Същата процедура може да се направи за всеки друг персонаж или обект, който трябва да взаимодейства с тревата.

В активната камера трябва да се деактивира дисплей слой 17, така че обектите, които могат да взаимодействат само с `SimpleGrassTexture`, да не се виждат.

Във функцията `_process` или `_physics_process` на персонажа трябва да се актуализира `SimpleGrassTextured` чрез извикване на `SimpleGrass.set_player_position(global_position)`.

Приставката позволява използването на 3D мрежи, конфигурирани от потребителя като различни типове цвята и др.

Ако е активиран интерактивния режим, времето за зареждане може да бъде ускорено чрез „изпичане“ на картата на височината (главно ако има голям брой треви) в менюто `SimpleGrassTextured` в горната лента.

Заклучение

Описаните решения за създаване на ресурси, подпомагащи автоматичното създаване на обкръжението в 3D игровия дизайн в помощ на обучението са полезни при оптимизиране на работата при разработки, създавани от малък екип, на база автоматизация на генерираното съдържание. Използването на подобни решения с отворен код са изключително полезни при създаването на триизмерно съдържание за обучението, особено когато разработваният проект не е финансиран, или е с ограничено финансиране.

Като недостатък може да се посочи, че такива решения с отворен код се създават от независими автори и не е гарантирана тяхната поддръжка за предстоящи нови издания на игровия двигател, за който са предназначени. Това може да затрудни използването им при актуализации на игровия двигател, да доведе до несъвместимости и необходимост от индивидуална адаптация и промени в кода, което понякога може да бъде трудно постижимо без познание на принципите и алгоритмите за работа на съответното решение.

ИЗПОЛЗВАНА ЛИТЕРАТУРА

Бурова, И. (2022). Приложение на интерактивни практики в часовете по технологии и предприемачество. Условия за провеждане и резултати от експеримента, Сборник научни трудове „Компетентностният подход като алтернатива пред предизвикателствата на 21. век“, 2022, ISBN 978-619-201-677-7, с. 75-83.

Буров, И. (2024). Базови ресурси, подпомагащи автоматичното създаване на обкръжението в 3d игровия дизайн в помощ на обучението, Годишник на шуменския университет „Епископ Константин Преславски“, Педагогически факултет, Т. XXVIII D, Шумен 2024 г., ISSN: 1314 – 6769, с. 427-435 и в Е-списание "Образование и развитие" ISSN 2603-3577, бр.13 2024 г. с. 147-157.

Буров, И. (2022). Изграждане на дигитална среда за интерактивно 3D игрово обучение. Разработка и интеграция на интерактивни 3D решения. Университетско издателство „Епископ Константин Преславски“ 2022 г. ISBN 978-619-201-661-6.

Буров, И. (2021). Подбор на игрови двигатели при разработка на 3D интерактивно съдържание в обучението, Годишник на ШУ „Еп. Константин Преславски“, т. XXV D, 2021, с 335-345.

Crawford, P. Godot Road Generator. Достъпно на: <https://github.com/TheDuckCow/godot-road-generator> [прегледано на 2025-04-15].

SimpleDungeons. <https://github.com/majikayogames/SimpleDungeons> [прегледано на 2025-04-15].

Frandsen K. Waterways Add-on for Godot Engine. Достъпно на: <https://github.com/Arnklit/Waterways> [прегледано на 2025-04-15].

IcterusGames cave_demo. https://github.com/IcterusGames/cave_demo [прегледано на 2025-04-15].

*проф. д-р Ивайло Иванов Буров
Шуменски университет „Епископ Константин Преславски“
Педагогически факултет
катедра „Педагогика и управление на образованието“
e-mail: i.burov@shu.bg*