Документација по предметот Тимски Проект

Создавање и имплементација на паркур систем базиран на издржливост (stamina) во 2Д платформер игра во Godot.

Членови**:**

Мартин Јаневски - 185016

Симон Мирчевски - 181207

Филип Стефановски – 186102

1. Информации за софтверот

Пред да се започне со документација на самата игра, во прилог има неколку информации за самиот софвер Godot и кои се сите бенефити од негово користење:

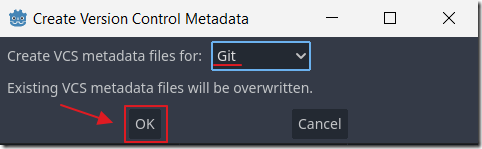
* Godot е целосно бесплатен за користење, без лиценцни такси или провизии што го прави достапен за инди девелопери во мали тимови како нашиот.
* Самиот софтвер е лесен, со мала големина, и може да работи на различни платформи. Оваа ефикасност овозможува развој на игри без потреба од моќен хардвер. Дури и на некои помали проекти може да се променува кодот симултано со пуштена инстанца од игра во позадина.
* Godot има доста корисни додатни класи и алатки за создавање на 2Д игри што би биле споредливи со системот развиен во Unity. Исто така, поддржува развој на 3Д игри иако не е толку напреден во споредба со 2Д.
* Во Godot секој елемент во играта е сведен на структура наречена јазол (node), и овие јазли можат лесно да се комбинираат и организираат за да се создадат сцени и нивоа. Ова ја намалува комплексноста на самите игри – доколку се промени кодот во еден јазол нема да влијае на другите компоненти од играта.
* Godot користи свој јазик за креирање на скрипти наречен GDScript - сличен на Python. Самиот јазик е лесен за учење и нема многу правила кои треба да се запазат. Исто така ги поддржува и C#, C++ и VisualScript за тие што преферираат други програмски јазици.
* На игри креирани со Godot може да им се направи експорт во повеќе платформи вклучувајќи ги Windows, macOS, Linux, Android, iOS, и веб прелистувачи (доколку сакате да создадете веб-базирана игра). Самата интеграција со овие платформи лесно се имплементира во игрите, дури има и детални туторијали за целиот процес.
* Godot има активна заедница која придонесува за неговиот развој, обемна документација и голем број на онлајн туторијали, што го прави уште полесен за учење и за добивање на поддршка доколку се заглави во создавањето на сегмент од игрите.
* За разлика од софтвери како Unreal, Godot не наплаќа провизии на игрите што ги креирате, овозможувајќи повеќе финансиска слобода за девелоперите.
* Godot добива чести ажурирања и подобрувања од заедницата, при што со секоја нова верзија има некоја нова корисна алатка што го упростува процесот на правење на игри.

Сумаризирано во кратки црти, во споредба со други софтвери за создавање на игри како Unity и Unreal, Godot се истакнува во развојот на 2D игри и нуди поголема флексибилност со помали трошоци - што го прави посебно привлечен за мали тимови и инди девелопери.

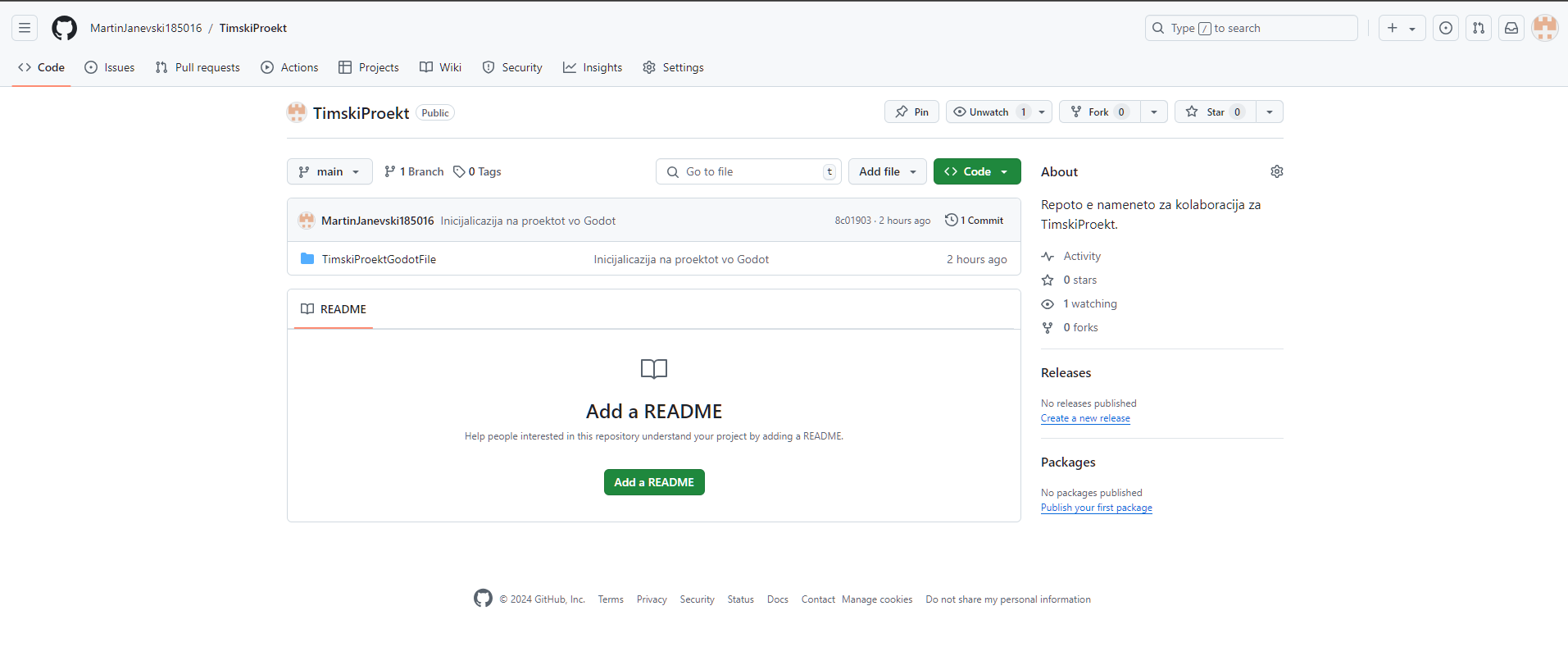
1. Верзионирање и изработка во тим

Годот има вграден систем за верзионирање и зачувување на развојот на самите игри/кодот во скриптите преку Git.

При креирање на самиот проект може да се имплементира верзионирање преку самото IDE на Годот. Истиот проект преку оваа функционалност беше прикачен на GitHub за колаборација со другите членови тимот.



На следниов [линк](https://blazorhelpwebsite.com/ViewBlogPost/15069) има корисен туторијал со кој што може да се додаде алатка за пуштање на верзии на кодот кон репозиториумот на Github, иако некои членови преферираа директно користење на Git Bash наспроти интерфејс.



*Почетниот изглед на репото при креирање на празен проект во Годот*

1. План за создавање на игра во Годот

Планот за содавање на играта е поделен во неколку главни сегменти карактеристични на платформер игри:

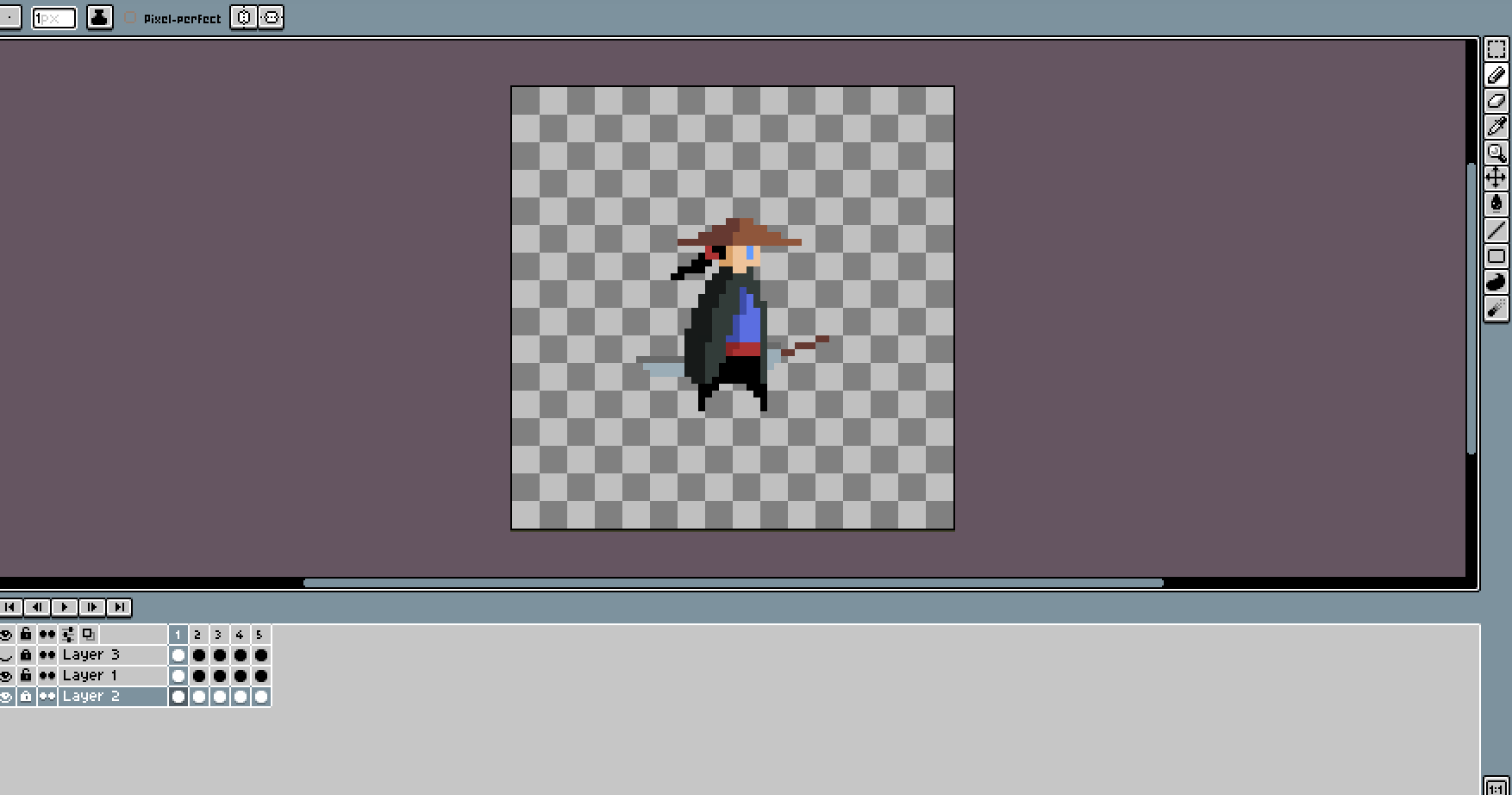
1. Создавање на функционален главен карактер со мапирани движења од типот:
   1. Движење во сите насоки (со додадена физика на успорување што би го направило природно)
   2. Скок
   3. Двоен скок
   4. Скок во воздух при паѓање од платформа (coyote jump)
   5. Скок од ѕид и одбивање од ѕидови
   6. Лизгање на зид
   7. Спринт (нагло движење на земја за одбегнување на стапици)
   8. Клекнување под платформи и движење во клекната позиција
2. Создавање на нивоа со кои ќе се прикажат истите движења на карактерот:
   1. Физички тела кои треба да се избегнат со скок/двоен скок или скок од ѕид
   2. Тела кои треба да се поминат со клекнување или користење на спринт
   3. Проектили кои треба да се одбегнат со временско планирање на скок/лизгање од ѕид или користење на спринт
   4. Создавање на препреки кои ќе го натераат карактерот да изгуби во играта
3. Создавање на систем за стамина на карактерот

По создавањето на самите движења на карактерот и нивоата во играта би дошол поволниот момент за создавање на таканаречен ‘stamina’ систем или со други зборови систем на издржливост на движење на карактерот. Овој систем би го лимитирал бројот на движења што може да ги направи играчот во одреден временски период со цел паметно да ги планира движењата и потешки да се препреките на играчот.

1. Создавање на карактерот и позадините

Сите позадини, препреки и анимации поврзани со карактерот се нацртани од страна на Мартин Јаневски во програма наречена Asperite.

Asperite е програма наменета специфично за цртање и анимација на карактери и позадини во видеоигри со 8-бит/пикселиран изглед.

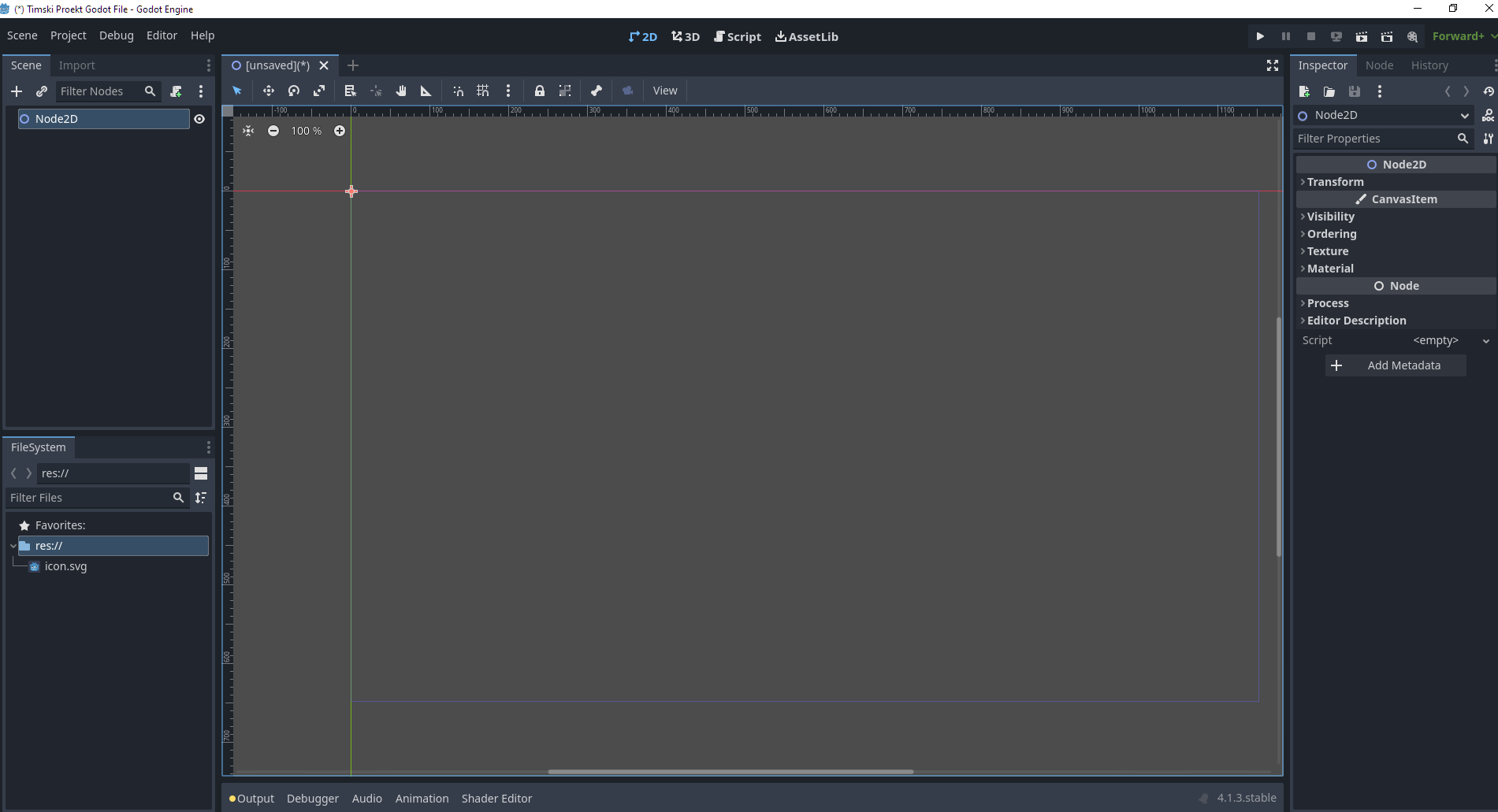


*Пример од интерфејсот на апликацијата Asperite*

5. Документација на процесот на креирање на играта

1. Создавање на главните сцени во Годот

Започнуваме со создавање на нов празен проект во Годот, при што непроменетиот интерфејс со 2Д јазол ни изгледа како на наведената слика:

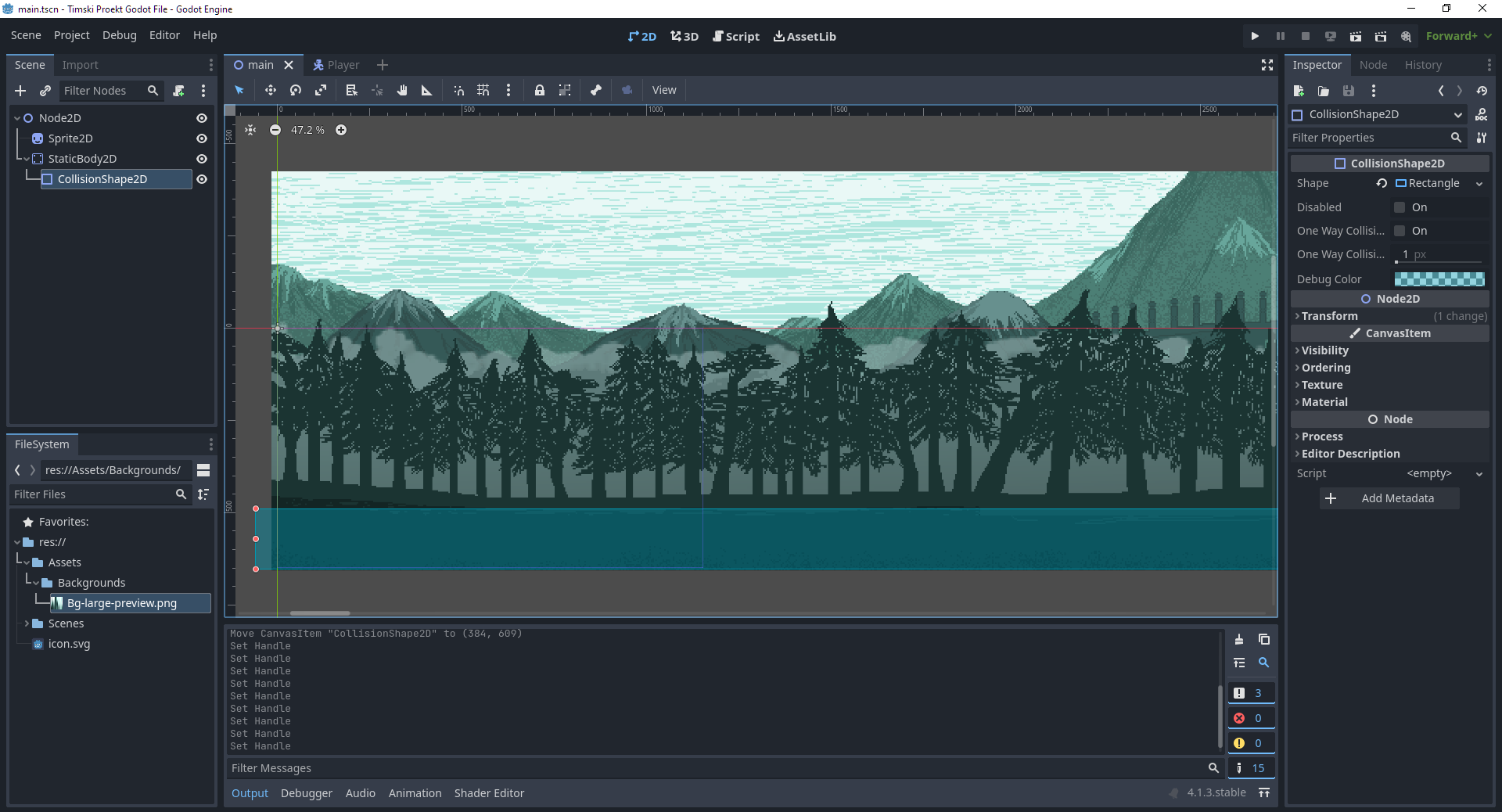


Додаваме две различни сцени, едната за самото ниво во играта (main scene) во кое ќе се придвижува карактерот и друга за создавање на самиот карактер (Player scene).

Во иднина ќе го инстанцираме карактерот во нивото за играта и така ќе проверуваме секоја функционалност на карактерот дали работи како испланирано.

Во сцената за нивото додаваме два јазли од тип Sprite2D (јазол со слика за додавање на позадина) и StaticBody2D (јазол за додавање на под со интегрирана физика) кој понатаму ќе има интеракција со карактерот. Потоа на тој StaticBody2D јазол ќе му додадеме CollisionShape2D јазол во форма на правоаголник кој ќе го покрива долниот дел на сликата и ќе претставува под на играта.

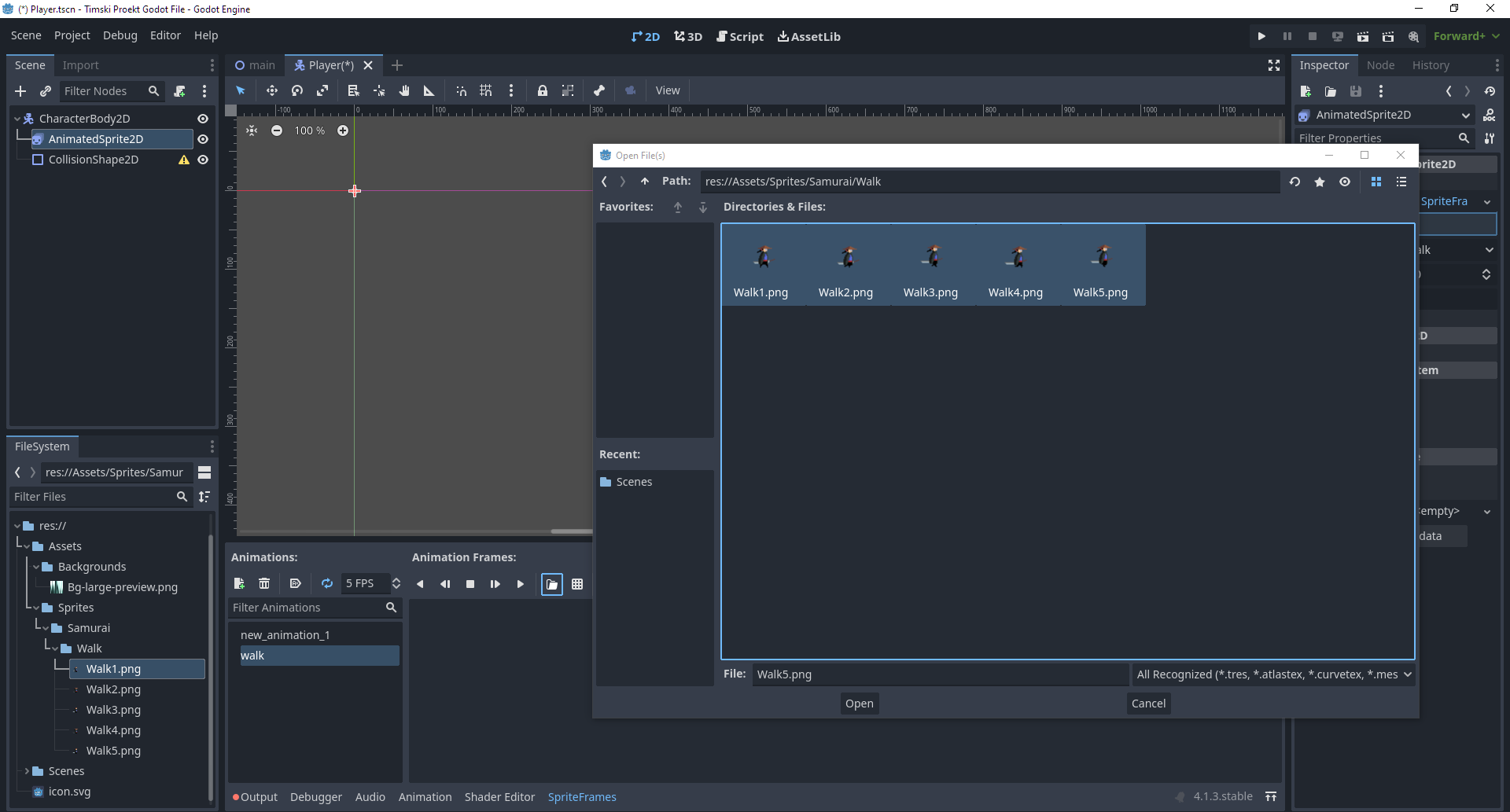
Откако ќе ги додадеме формата на нашиот под и позадинската слика во фолдер за позадини сцената ќе ни изгледа како наведено:



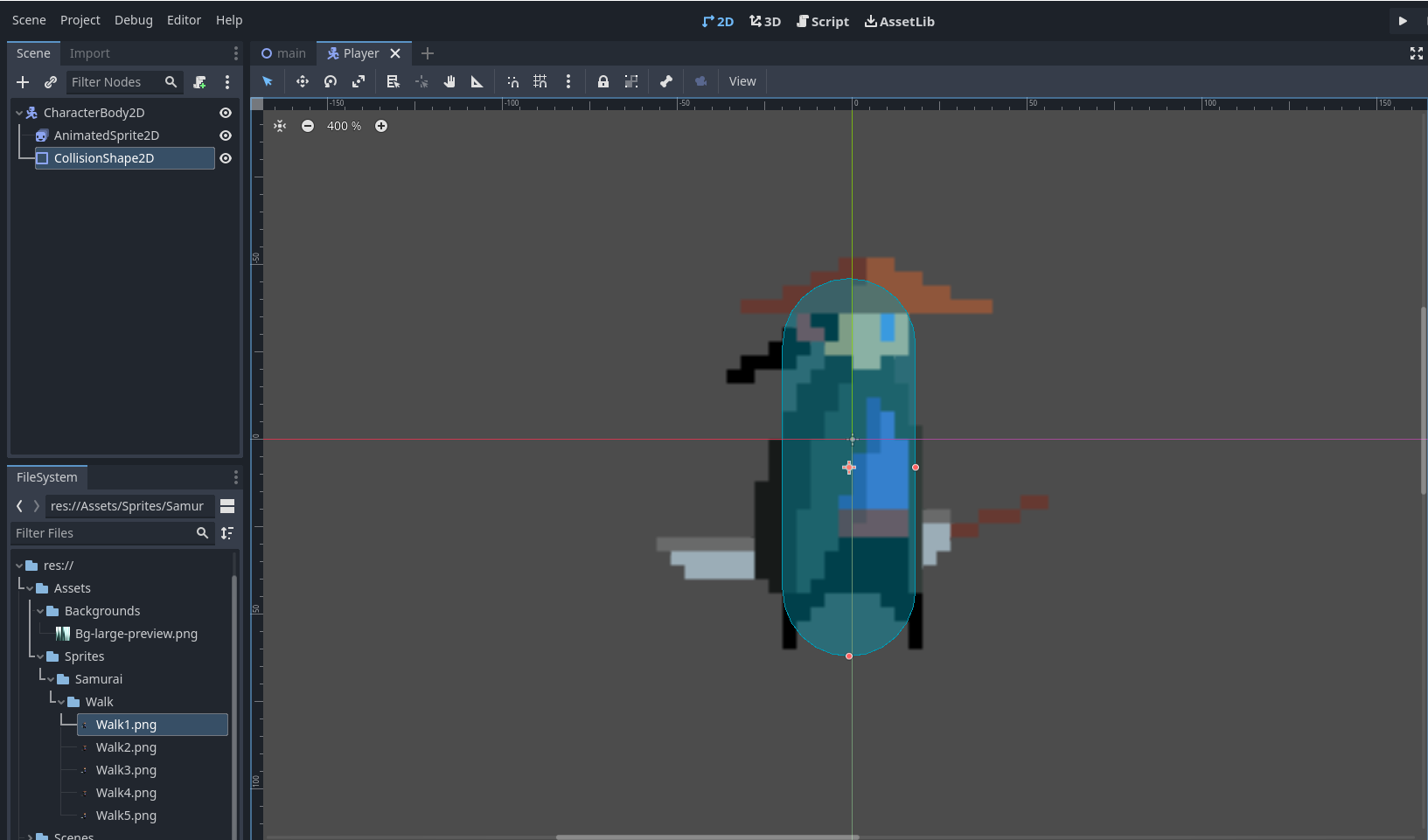
За сцената со играчот додаваме неколку типови на јазли со различни функционалности:

* CharacterBody2D – има своја почетна физика и гравитација и е наменет за карактери кои ќе ги контролира самиот играч
* AnimatedSprite2D – се користи за интеграција на анимации со движења предизвикани од тастатура
* CollisionShape2D – физичка форма која ја користи карактерот за интеракција со други површини

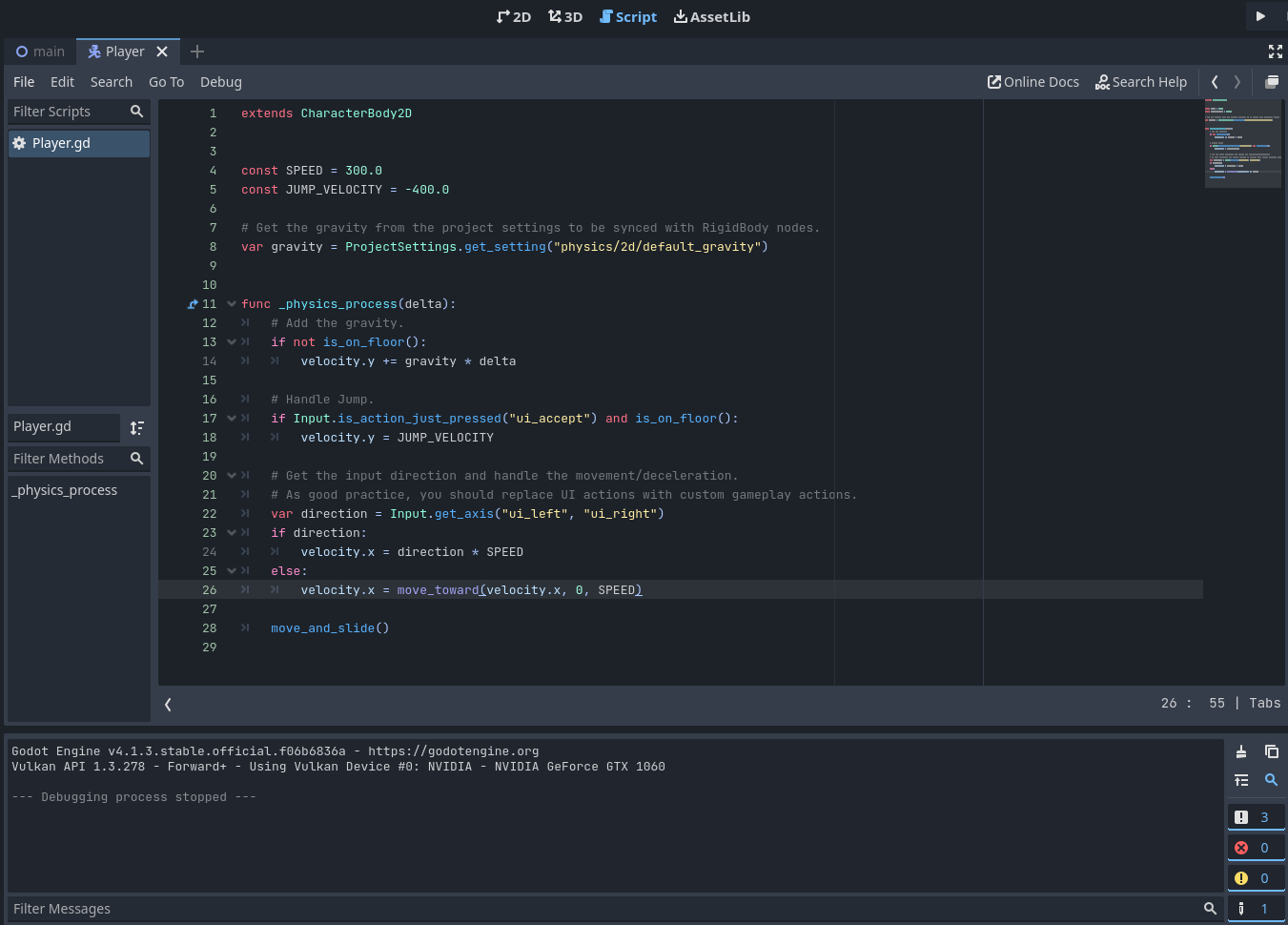
Ја додаваме формата на карактерот (во нашиов случај капсула) и анимациите за одење на карактерот (ставени во посебен фолдер) во играта. За да го направиме тоа треба во AnimatedSprite2D јазолот да одбереме SpriteFrames како избор на тип на анимација и да додадеме нова анимација со име walk. Во таа анимација ќе ги импортираме сликите од нашиот карактер и ќе ги наместиме по редослед (прикажано на сликата).



Сцената на карактерот после сите овие промени е прикажана на сликата:



За да започнеме со кодирање на движењата на самиот карактер мора да креираме нова скрипта и да ја поврземе со него. При креирање на оваа скрипта Годот има почетен код за скок и движење на тој CharacterBody2D јазол.



*Пример приказ за Скрипта во Годот*