Szegedi Tudományegyetem

Informatikai Intézet

**SZAKDOLGOZAT**

Hallgató:

Török Dániel

2025

Szegedi Tudományegyetem

Informatikai Tanszékcsoport

2D souls-like platformer játék a Godot játék motorban

Szakdolgozat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Készítette: |  | Témavezető: |  |
|  | Török Dániel |  | Jász Judit dr. |  |
|  | programtervező informatikus szakos hallgató |  | adjunktus |  |

Szeged

2025

FELADATKIÍRÁS

A szakdolgozat célja egy 2D-s platformer souls-like videójáték fejlesztése, amelyben a játékos egy nyitott világban tud felfedezni, illetve ellenségekkel küzdeni, miközben fejleszti a karakterét, tárgyakat gyűjt és készít, képességeket old fel, amikkel később további részeket old fel a nyitott világból. Fontos, hogy a játék állást el lehessen menteni. továbbá betölteni.

Tartalmi összefoglaló

* ***A téma megnevezése***

2D souls-like platformer játék a Godot játék motorban

* ***A megadott feladat megfogalmazása***:

Egy olyan videójáték fejlesztése, amely lehetővé teszi a játékos számára, hogy fejlessze a karakterét, tárgyakat készítsen, ellenségekkel küzdjön, továbbá a játék tartalmazzon platformer elemeket (pl.: fal ugrás, dupla ugrás, platformok), a karakter halálakor a világban levő ellenfelek álljanak vissza (kivétel a főellenség(-ek)), egy rendszer, amely lehetővé teszi a játékos számára, hogy teleportálhasson előre meghatározott helyekre. Fontos funkciók fejlesztése példáúl a játék állás mentése, játék betöltése.

* ***A megoldási mód:***

Az játék fejlesztése Godot játék motor segítségével történt. A pixeles képek megrajzolásához a Pixeloroma program lett használva, audió fájlok a freesound.org webhelyről lettek beszerezve, verzió kezeléshez a github.com lett használva. A pálya, illetve minden más egyéb fájl a játék motoron belül lett elkészítve.

* ***Alkalmazott eszközök, módszerek:***

Pixelorama

Godot engine 4.2, 4.3, 4.4, 4.5

Visual Studio Code fejlesztői környezet

GDScript beépített programozási nyelv

* ***Elért eredmények:***

Az elkészült játék lehetővé teszi a játékos egy előre elkészített pálya felfedezését, amely több részre van bontva, közben ellenségekkel küzdjön meg beleértve a főllenség(ek)-et, csapdákat kerüljön ki, fejlessze karakterét, tárgyakat hozzon létre/ gyűjtsön össze, elmentse a játék állását, betöltse a játék állását, teleportálhasson előre meghatározott pozíciókra. Az ellenfelek visszaállításra kerülnek a játékos halálakor. Platformer elemek implementálásra kerültek.

* ***Kulcsszavak:***

Godot, GDScript, souls-like, souls, platformer, tárgyak, játék, videojáték, nyitott világ

Tartalomjegyzék

[FELADATKIÍRÁS 1](#_Toc211296930)

[Tartalmi összefoglaló 2](#_Toc211296931)

[Tartalomjegyzék 4](#_Toc211296932)

[BEVEZETÉS 5](#_Toc211296933)

[1. Technológiák bemutatása 5](#_Toc211296934)

[1.1. Programnyelvek, technológiák 5](#_Toc211296935)

[1.1.1. A Godot engine bemutatása 5](#_Toc211296936)

[1.1.2. A GdScript programozási nyelv 6](#_Toc211296937)

[1.1.3. A jelenet- és csomópont alapú architektúra 7](#_Toc211296938)

[1.1.4. A fizikai rendszer és animációkezelés 7](#_Toc211296939)

[1.1.5. Exportálás és platformfüggetlenség 7](#_Toc211296940)

[1.2. Fejlesztői környezet és eszközök 7](#_Toc211296941)

[1.3. A 2D játékfejlesztés sajátosságai 8](#_Toc211296942)

[1.4. A souls-like műfaj technikai jellemzői 9](#_Toc211296943)

[Irodalomjegyzék 11](#_Toc211296944)

BEVEZETÉS

A videojáték-fejlesztés az elmúlt évtizedekben az informatikai ipar egyik legdinamikusabban fejlődő területévé vált. A játékkészítés mára nem csupán szórakoztatóipari tevékenység, hanem komplex szoftverfejlesztési folyamat, amely ötvözi a programozást, a grafikai tervezést, a hangdizájnt és a mesterséges intelligenciát. A játékfejlesztő eszközök és motorok fejlődése lehetővé tette, hogy kis fejlesztőcsapatok vagy akár egyéni alkotók is professzionális minőségű játékokat hozzanak létre.

A szakdolgozat célja egy 2D, úgynevezett „souls-like” platformer játék megvalósítása a Godot Engine segítségével. A „souls-like” kifejezés olyan játékstílust jelöl, amelyet a Dark Souls sorozat inspirált: a jellemzői közé tartozik a magas nehézségi szint, a pontos időzítést igénylő harcrendszer, valamint a fokozatosan felfedezhető, összefüggő játéktér. A cél egy olyan prototípus létrehozása, amely hűen visszaadja ennek a műfajnak a hangulatát és mechanikai elemeit, ugyanakkor bemutatja a Godot Engine fejlesztési lehetőségeit és hatékonyságát.

A választás indoka kettős: egyrészt a Godot Engine nyílt forráskódú és szabadon felhasználható, így ideális környezet az oktatási és kísérleti célú fejlesztésekhez; másrészt a souls-like műfaj összetett játékmenete kiváló lehetőséget biztosít a különböző játékrendszerek — például harc, mesterséges intelligencia, fizika és felhasználói interfész — integrálásának bemutatására. A projekt során a hangsúly a játékmenet logikai felépítésén, a karaktervezérlésen, az ellenségek viselkedésén és a pályarendszer kialakításán lesz.

A dolgozat a fejlesztési folyamat elméleti és gyakorlati aspektusait egyaránt tárgyalja. Az első fejezet a Godot játékmotor működését, felépítését és programozási környezetét mutatja be. Ezt követően részletesen ismertetésre kerül a játék tervezése, az implementáció folyamata, valamint az alkalmazott technológiák. A záró fejezet a fejlesztés során szerzett tapasztalatokat, valamint a lehetséges továbbfejlesztési irányokat foglalja össze.

1. TECHNOLÓGIÁK BEMUTATÁSA

1.1. Programnyelvek, technológiák

1.1.1. A Godot engine bemutatása

A **Godot Engine** egy modern, nyílt forráskódú játékmotor, amelyet Juan Linietsky és Ariel Manzur fejlesztett ki, és 2014-ben jelent meg az első stabil változata. A motor célja egy egységes, platformfüggetlen fejlesztői környezet biztosítása 2D és 3D játékok készítéséhez. A Godot alapja **C++ nyelven** íródott, és **GDScript** nevű, kifejezetten a játékfejlesztéshez tervezett, magas szintű szkriptnyelvet használ. A projektet a **Godot Foundation** és egy aktív közösség tartja fenn, amely folyamatos frissítésekkel és kiegészítésekkel biztosítja a rendszer fejlődését és hosszú távú támogatását.

A Godot a **jelenet- és csomópontalapú architektúrára** épül, amely moduláris felépítést és magas fokú újrafelhasználhatóságot tesz lehetővé. A motor beépített komponensei között megtalálható a **fizikai rendszer**, az **animációkezelés**, a **vizuális shaderrendszer**, valamint a **felhasználói felületek kialakítását** segítő eszköztár. A fejlesztők több szkriptnyelv közül választhatnak, így a GDScript mellett **C# és C++** is használható, ami megkönnyíti a különböző fejlesztési igényekhez való alkalmazkodást.

A Godot Engine előnyei közé tartozik, hogy **teljesen platformfüggetlen**, és támogatja a játékok exportálását **Windows, Linux, macOS, Android, iOS és Web (HTML5)** rendszerekre. A motor jól integrálható külső eszközökkel, például **Blenderrel, REST API-kkal vagy adatbázisokkal**, továbbá lehetőséget nyújt **VR- és AR-fejlesztésekre**, valamint **multiplayer rendszerek** kialakítására is.

A Godot választása azért indokolt, mert egy **ingyenesen elérhető, átlátható és professzionális fejlesztőkörnyezetet** kínál, amely magas fokú rugalmasságot biztosít. A hivatalos dokumentáció részletes útmutatókat és példákat tartalmaz, míg az aktív fejlesztői közösség fórumokon, blogokon és közösségi platformokon keresztül folyamatosan bővíti a tudásbázist. A motor nyílt forráskódjának köszönhetően a fejlesztők teljes kontrollt gyakorolhatnak a projekt felett, ami különösen fontos a kutatás-fejlesztési és oktatási célú felhasználás során.

1.1.2. A GdScript programozási nyelv

A GDScript a Godot Engine saját, magas szintű programozási nyelve, amelyet kifejezetten a játékfejlesztés megkönnyítésére terveztek. Szintaxisa a Python nyelvre emlékeztet, így könnyen elsajátítható, ugyanakkor teljes mértékben a Godot architektúrájához igazodik. A GDScript szorosan integrált a motor komponenseivel, ezért közvetlenül hozzáfér a jelenetekhez, csomópontokhoz és a motor funkcióihoz. A nyelv dinamikusan típusos, de támogatja az opcionális típusmegadást is, ami növeli a kód biztonságát és átláthatóságát. A GDScript használata lehetővé teszi a gyors prototípus-készítést, ugyanakkor alkalmas komplex logikai rendszerek megvalósítására is. Mivel közvetlenül a motorhoz készült, futása optimalizált és hatékony, így ideális választás a Godot-alapú fejlesztésekhez.

1.1.3. A jelenet- és csomópont alapú architektúra

A Godot Engine központi koncepciója a jelenet- és csomópontalapú rendszer, amely a fejlesztés modularitását és rugalmasságát biztosítja. Minden elem — legyen az egy karakter, kamera vagy fizikai objektum — egy csomópont (Node) formájában jelenik meg, amely hierarchikus struktúrában szerveződik. Ezekből a csomópontokból épülnek fel a jelenetek (Scenes), amelyek önálló egységként kezelhetők, majd más jelenetekbe ágyazhatók. Ez a megközelítés megkönnyíti a komponensek újra felhasználását, a projektek áttekinthetőségét és a csapatmunkát. A hierarchikus modell segítségével a fejlesztők hatékonyan tudják kezelni a játéklogikát, az animációkat és a felhasználói felületet egyaránt.

1.1.4. A fizikai rendszer és animációkezelés

A Godot beépített fizikai motorral rendelkezik, amely támogatja a 2D és 3D objektumok mozgását, ütközését és dinamikus kölcsönhatásait. A motor külön kezeli a 2D és 3D fizikai rendszereket, így mindkettő optimalizált teljesítményt nyújt. Az animációrendszer szintén fejlett: kulcsképkockás, görbékkel vezérelt, valamint csontvázalapú animációkat is kezel. Az AnimationPlayer és AnimationTree komponensek lehetővé teszik a komplex mozgások és átmenetek precíz irányítását. A Godot vizuális szerkesztője révén az animációk idővonal-alapon hozhatók létre, így a fejlesztés folyamata intuitív és vizuálisan jól követhető.

1.1.5. Exportálás és platformfüggetlenség

A Godot Engine egyik legnagyobb előnye a platformfüggetlenség. A kész játékok egyszerűen exportálhatók több operációs rendszerre, beleértve a Windows, Linux, macOS, Android, iOS és Web (HTML5) környezeteket. Az exportálás folyamata automatizált, és a beállítások minden platformhoz külön testreszabhatók. A Godot támogatja a Progresszív Webalkalmazások (PWA) és a mobilalkalmazások fejlesztését is, valamint integrálható külső szolgáltatásokkal, például adatbázisokkal, REST API-kkal vagy felhőalapú megoldásokkal. Ez a rugalmasság lehetővé teszi, hogy a fejlesztők egyetlen forráskódból több eszközre is hatékonyan publikáljanak.

1.2. Fejlesztői környezet és eszközök

A játék fejlesztéséhez két fő szoftveres eszközt alkalmaztam: a Godot Engine-t és a Pixelorama pixelgrafikai szerkesztőt. A Godot Engine a fejlesztés központi eleme, amely a játék logikájának implementálását, a jelenetek kezelését, a fizikai rendszert, az animációk kezelését, valamint a felhasználói felület kialakítását teszi lehetővé. A Godot integrált fejlesztői környezete (IDE) lehetővé teszi a kód és a vizuális elemek egyidejű kezelését, valamint a játékok egyszerű exportálását különböző platformokra. A motor beépített eszközei — például a SceneTree, az AnimationPlayer és a TileMap komponensek — lehetővé teszik a moduláris felépítést és az objektumok hierarchikus szervezését, ami különösen fontos egy összetett játékmenet implementálásánál.

A vizuális elemek előállításához a Pixelorama programot használtam, amely egy nyílt forráskódú, 2D pixelgrafikai szerkesztő. A program lehetővé teszi sprite-ok, háttérképek és animációs képkockák készítését, amelyek közvetlenül importálhatók a Godot Engine-be. A Pixelorama egyszerű, intuitív felülete és a rétegkezelés támogatása megkönnyíti a játék vizuális elemeinek szerkesztését, míg az exportált képek formátuma kompatibilis a Godot által használt erőforrás-kezeléssel.

A két eszköz kombinációja lehetővé teszi a fejlesztés teljes folyamatának lefedését a vizuális tervezéstől a kód alapú logikáig, miközben a nyílt forráskódú jellegük biztosítja a szabad felhasználást és testreszabhatóságot. A fejlesztői környezet kiválasztása során kiemelten fontos szempont volt a könnyű integráció, a dokumentáció és a közösségi támogatás elérhetősége, amelyek a projekt hatékony megvalósítását segítik.

1.3. A 2D játékfejlesztés sajátosságai

A 2D játékfejlesztés során a játékterek, karakterek és interakciók kétdimenziós síkon valósulnak meg, ami jelentősen befolyásolja a játékmechanika és a programozás felépítését. A játék világát sprite-ok és tilemap-ek segítségével lehet modellezni, amelyek egységes koordinátarendszerben helyezkednek el, és a kamera mozgása határozza meg a játékos látóterét. A tilemap-ek alkalmazása lehetővé teszi a pályák moduláris felépítését, ami elősegíti a pályaszerkesztés hatékonyságát és az újrafelhasználhatóságot.

A 2D-s játékok esetében kiemelten fontos a rétegkezelés (z-index), amely meghatározza az objektumok egymáshoz viszonyított láthatóságát, valamint az ütközésdetektálás precíz implementálása. A fizikai kölcsönhatások — például a gravitáció, az akadályokkal való ütközés és a karakter mozgása — a 2D-s sík sajátosságaihoz igazodnak, így optimalizált algoritmusokkal és beépített fizikai komponensekkel valósíthatók meg.

A 2D játékfejlesztés további jellemzője a sprite-alapú animációk kezelése, amelyek kulcsképkockák és animációs ciklusok formájában kerülnek implementálásra. A karakterek mozgása, az ellenségek viselkedése, valamint a vizuális effektek szoros kapcsolatban állnak a játék logikájával, ami különösen fontos egy souls-like jellegű platformer esetében, ahol a pontosság és a játékos interakcióinak visszajelzése alapvető élményelem.

Összességében a 2D játékfejlesztés sajátosságai megkövetelik a vizuális és a logikai komponensek szoros integrációját, valamint a játékmechanikák precíz, koordinátarendszer-alapú megvalósítását. A Godot Engine és a Pixelorama eszközkombinációja lehetővé teszi ezen követelmények hatékony teljesítését, biztosítva a játék funkcionális és vizuális konzisztenciáját.

1.4. A souls-like műfaj technikai jellemzői

A „souls-like” műfaj olyan játékmeneti elemeket foglal magában, amelyek a Dark Souls sorozatból váltak ismertté. Ezek közé tartozik a nagy nehézségi szint, a precíz időzítést igénylő harcrendszer, valamint a fokozatosan felfedezhető, összefüggő pályarendszer. A technikai megvalósítás szempontjából ezek az elemek jelentős kihívást jelentenek, különösen egy 2D platformer játék esetében.

A harcrendszer megvalósításához fontos a pontosan definiált ütközési mezők (hitboxok) és a karakterek mozgásának finomhangolása. A támadások, védekezések és mozdulatok időzítése kritikus szerepet játszik a játék élményében, ezért a mozgások és animációk szoros szinkronban kell, hogy legyenek a játék logikájával. A támadás- és védekezésmechanizmusok implementálása során a fejlesztőnek figyelembe kell vennie a frame-alapú időzítést és az esetleges invincibility frame-ek kezelését, amelyek a játékos és az ellenségek interakcióit szabályozzák.

Az ellenségek viselkedése és mesterséges intelligenciája (AI) a souls-like játékok kulcseleme. Az AI lehetővé teszi az ellenségek adaptív és kihívást jelentő reagálását, például a támadások előrejelzését, a távolságtartást, vagy a pályaelemekhez való alkalmazkodást. A 2D környezetben az AI implementálása magában foglalja a mozgás útvonalak definiálását, a látómező és a közelharci távolság figyelembevételét, valamint a különböző támadásminták kialakítását.

A souls-like játékok további jellegzetessége a checkpoint- és mentési rendszer, amely a játékos halálát követően biztosítja a folytonosságot, ugyanakkor megtartja a kihívás élményét. Emellett a pályák összefüggő struktúrája és a titkos útvonalak, visszatérő helyszínek kialakítása kiemelt figyelmet igényel a játéktér modellezésében.

2. SPECIFIKÁCIÓ

A következő fejezet célja a fejlesztett játék részletes specifikációjának bemutatása. Ebben a részben kerülnek ismertetésre a játék alapvető céljai, mechanikái, felépítése, valamint a működését meghatározó technikai és tervezési döntések. A fejezet kitér a játékmenet fő elemeire, a vizuális és interaktív komponensekre, illetve arra, hogyan valósulnak meg ezek a Godot Engine környezetében.

A specifikáció részletes leírása alapot biztosít a későbbi implementációs folyamat megértéséhez, valamint meghatározza azokat a követelményeket és jellemzőket, amelyek mentén a játék fejlesztése és tesztelése megvalósult.

2.1. Játékmenet

A fejlesztett játék egy kétdimenziós, oldalnézetes souls-like platformer, amely egy nagy, összefüggő, nyitott pályán játszódik. A játékos szabadon felfedezheti a világ különböző területeit, miközben ellenségekkel küzd, új útvonalakat tár fel, és fokozatosan fejlődik. A játékmenet központi eleme az új helyszínek és titkos átjárók felfedezése, amelyek gyakran új ellenségeket, kihívásokat és jutalmakat rejtenek. A felfedezés folyamatos jutalmazása fenntartja a játékos motivációját, és lehetőséget ad a saját útvonal megválasztására.

A játék során a játékos egyetlen, univerzális erőforrást gyűjt, amelyet minden fontos tevékenységhez felhasználhat. Ezzel az erőforrással lehet fejleszteni a karakter tulajdonságait, új felszereléseket vagy képességeket megszerezni, illetve bizonyos interaktív elemeket aktiválni. Az egységes erőforrásrendszer tudatos döntés elé állítja a játékost: kockáztasson és tovább haladjon több gyűjtött egységgel, vagy használja fel azokat biztonságos pontokon a fejlődés érdekében. Ez a mechanika a souls-like játékokra jellemző kockázat-jutalom elv egyik legfontosabb eleme.

A harcrendszer a műfaj sajátosságait követi: minden támadásnak, ugrásnak és védekezésnek súlya és időzítése van. A játékosnak fel kell ismernie az ellenségek mozgásmintáit, és ezek alapján kell reagálnia. A halál a játékmenet szerves része — a játékos ilyenkor visszakerül a legutóbbi aktivált ellenőrzőpontra, és elveszíti a nála lévő erőforrásokat, de lehetőséget kap azok visszaszerzésére. Ez a ciklikus előrehaladás és újrapróbálás adja a játék egyik legfontosabb ritmusát és feszültségét.

2.2. Felhasználói felület

A játék felhasználói felülete (UI) minimalista, mégis funkcionálisan gazdag kialakítást követ, amely a *souls-like* játékokra jellemző letisztult vizuális stílust idézi. A cél a játékos teljes bevonása a játékmenetbe, ezért a UI elemek nem tolakodóak, kizárólag a legszükségesebb információkat jelenítik meg. A felhasználói felületet több, különböző funkciót betöltő menürendszer alkotja, amelyek mind a Godot engine saját felületkezelő komponenseire épülnek.

2.2.1. Kezdőképernyő

A játék indításakor a játékos a kezdőképernyőre kerül, ahol elérhető a legfontosabb főmenü-elemek: **Új játék**, **Betöltés**, **Beállítások** és **Kilépés**. A háttérben egy statikus, atmoszférikus háttérkép vagy animált jelenet fut, amely illeszkedik a játék hangulatához.  
A *Load* (betöltés) menüben megjelennek a korábban mentett állások, vizuális előnézettel (pl. karakter szintje, helyszín neve, mentési időpont). A mentési rendszer manuális mentést nem engedélyez a játékon belül, kizárólag a pihenőpontok (*checkpoints*) aktiválásakor történik automatikus mentés.

2.2.2. Játékon belüli főmenü

A játékon belül a főmenü a **Pause** funkcióval érhető el, amely megállítja a játék folyamatát. Innen a játékos elérheti a **Beállítások**, **Főmenübe vissza**, **Kilépés**, valamint a **Fejlődés** és **Crafting** menüket.

A menü navigációja egyszerű, egységesen a billentyűzet és kontroller támogatását is biztosítja, a Godot Control csomópontjain alapuló hierarchikus felépítéssel.

2.2.3. Beállítások menü

A beállítások menü a játékos számára lehetővé teszi az alapvető rendszerbeállítások módosítását, mint a **hangerő**, **felbontás**, **billentyűkiosztás**, valamint a **teljes képernyős mód**. A felület logikusan strukturált, a változtatások azonnal alkalmazhatók, és a beállításokat a játék automatikusan elmenti.

2.2.4. Fejlesztési menü

A fejlődés menü közvetlenül a *souls-like* játékok karakterfejlesztési rendszerét idézi. A játékos itt költheti el a gyűjtött erőforrásait különböző attribútumokra, mint például **életerő**, **állóképesség**, **sebzés** vagy **gyorsaság**. A fejlesztés kizárólag az *ellenőrző pontoknál* lehetséges, ezzel ösztönözve a taktikus döntéshozatalt.

A menü vizuálisan a klasszikus *Dark Souls* stílust követi: sötét tónusú háttér, finom tipográfia, és letisztult statisztika-megjelenítés jellemzi. Az új értékek kijelölés után csak megerősítésre kerülnek véglegesen alkalmazásra, így elkerülhető a véletlen fejlesztés.

2.2.5. Barkácsolás menü

A crafting rendszer lehetővé teszi különböző tárgyak, fegyverek vagy fejlesztések készítését a játék során gyűjtött alapanyagok felhasználásával. A menü átlátható kategóriákba rendezi a recepteket, és valós időben mutatja, mely összetevők állnak rendelkezésre.

A fejlesztés és készítés folyamata a gyűjtött univerzális erőforráshoz kötődik, ami szoros kapcsolatot teremt a játékmenet és a gazdasági rendszer között. A játékos így minden döntésénél mérlegelni kényszerül: új tárgyat készít, vagy inkább karaktert fejleszt.

2.2.6. Ellenőrző pont menü

A *checkpoint* menü a játék egyik központi eleme, amely a pihenés és mentés funkcióját látja el. A játékos itt biztonságosan pihenhet, visszatöltheti életerejét és állóképességét, valamint fejlesztheti karakterét vagy készíthet tárgyakat is bizonyos esetekben.

A *checkpoint* aktiválásával a játék automatikusan ment, azonban minden normál ellenség újraéled, hasonlóan a *Dark Souls* játékok rendszeréhez. Ezzel a megoldással a játék megőrzi a feszültséget, miközben a fejlődés biztonságos pontjait is biztosítja.

2.2.7. HUD

A HUD a játékos számára egyik legfontosabb felület, aminek minimálisnak kell lennie, és minden szükséges információt közölnie kell a játékossal, többek közt interakciók, életerő fontos gyűjthető tárgyak, jelenlegi fegyver/képesség jelzése, olykor szituációtól függően változhat, akár a teljes felület. Azért is fontos mert a játékos ezt a felületet látja a legtöbbször.

Irodalomjegyzék

[1] <https://godotengine.org/>

[2] <https://docs.godotengine.org/en/stable/>

[3] <https://docs.godotengine.org/en/stable/classes/index.html#nodes>

[4] <https://docs.godotengine.org/en/stable/classes/index.html#resources>

[5] <https://orama-interactive.itch.io/pixelorama>

[6] <https://en.wikipedia.org/wiki/Godot_(game_engine)>

[7] <https://en.wikipedia.org/wiki/Soulslike>

[8] <https://gameprogrammingpatterns.com/>

[9] <https://gameprogrammingpatterns.com/singleton.html>

[10] <https://gameprogrammingpatterns.com/state.html>

[11] <https://freesound.org/>

[12] <https://firebase.google.com/docs/hosting/quickstart>

[13] <https://developer.mozilla.org/en-US/docs/Web/HTML>

[14] <https://www.w3schools.com/html/>

[15] <https://www.w3schools.com/angular/>