RĪGAS VALSTS TEHNIKUMS

DATORIKAS NODAĻA

Izglītības programma: Programmēšana

KVALIFIKĀCIJAS DARBS

3D šausmu puzles spēle "*Uncanny Spaces*"

Paskaidrojošais raksts 99 lpp.

Audzēknis: Emīls Blinds

Prakses vadītājs: Igors Litvjakovs

Nodaļas vadītājs: Normunds Barbāns

Rīga 2024

ANOTĀCIJA

Kvalifikācijas darbs ir pirmās personas 3D šausmu puzles spēle “Uncanny Spaces”. Tā ir spēle, kas paredzēta šausmu spēļu izbaudītājiem, kur mērķis ir atrast vajadzīgos priekšmetus, izvairīties no entītijām, kas uzbrūk spēlētājam, un atrast izeju, lai veiksmīgi izbēgtu no mapes un uzvarētu spēli. Spēle tika izstrādāta ar *Unity* spēļu dzini, *Microsoft Visual Studio* programmēšanas vidi, rakstot visu kodu C# programmēšanas valodā.

Kvalifikācijas darbs ietver ievadu, uzdevumu nostādni, prasību specifikāciju, uzdevuma nostādni, prasību specifikāciju, uzdevuma risināšanas līdzekļu izvēles pamatojumu, programmatūras produkta modelēšanas un projektēšanas aprakstu, datu struktūras aprakstu, lietotāja ceļvedi, nobeigumu un pielikumus. Kvalifikācijas darba ievadā ir aprakstīta spēles auditorija, un tirgus izpēte. Uzdevuma nostādnē ir norādīti uzdevumi, kurus programma būs nepieciešams veikt. Prasību specifikācija sastāv no ieejas un izejas informācijas, kā arī no visām sistēmas funkcionālajām un nefunkcionālajām prasībām. Uzdevuma risināšanas līdzekļu izvēles pamatojumā ir norādīti, kādi līdzekļi tiks izmantoti izstrādāšanai un kādiem nolūkiem tie tiek izmantoti. Programmatūras produkta modelēšanas un projektēšanas apraksts sastāv no sistēmas struktūra modeļa, kas ietver sistēmas arhitektūru un UML klašu diagrammau, un funkcionālās sistēma modeļa, kas satur datu plūsmu modeli. Datu struktūru aprakstā tiek parādīta datu relāciju shēma, kā arī datu tabulas un shēma, ar to relāciju aprakstiem. Lietotāja ceļvedī ir norādītas nepieciešamās sistēmas prasības aparatūrai un programmatūrai, sistēmas instalācija un palaišana, kā arī programmas apraksts, kur paskaidro, kā pareizi lietot sistēmu. Testa piemērā ir dots kā pareizi jāstrādā opciju izvēlnes navigācijai, spēles skaļuma konfigurācijai, un spēlētāja darbībai ar objektiem.

Kvalifikācijas darbs sastāv no 99 lappusēm, kurā ietilpst 51. attēli, 10 tabulas, 30 pielikumi. Pielikumi satur visus spēles skriptus.

ANNOTATION

Qualification work is a first person 3D horror puzzle game “Uncanny Spaces”. It’s a videogame, made for horror game genre fans, with the goal of finding the requried items, evading entities, which attack the player, and find the exit, to successfuly escape the map and win the game. Game was made using *Unity* game engine, *Microsoft Visual Studio* development environment, coding everything in C# coding language.

The qualification work includes an introduction, task statement, requirements specification, task solution methods justification, software product modeling and design description, data structure description, user guide, conclusion, and appendices. The introduction describes the game's audience and market research. The task statement specifies the tasks that the program needs to accomplish. The requirements specification includes input and output information, as well as all the functional and non-functional requirements of the system. The task solution methods justification indicates the tools that will be used for development and their purposes. The software product modeling and design description includes the system structure model, which covers the system architecture and UML class diagrams, and the functional system model, which contains the data flow model. The data structure description presents the data relationship schema, as well as data tables and schema with their relationship descriptions. The user guide outlines the necessary system requirements for hardware and software, system installation and launch, and a description of the program explaining how to use the system correctly. The test example provides guidance on how to correctly navigate the options menu, configure game volume, and interact with objects as the player.

The qualification work consists of 99 pages, including 51 images, 10 tables, and 30 attachments. The attachments contain all the game scripts.

SATURS

IEVADS

Mūsdienās pastāv daudz šausmu spēles, kā arī daudz puzles šausmu spēles, un daudzas no tām izpaužās katra ar savu elementu, piemērām, baismīgas entītijas, vai biedējošiem izbīļiem, bet nepastāv daudz šausmu spēļu, kas fokusē uz atmosfēras veidošanu, ko varētu apstrīdēt, ka ir pats galvenākais elements, kas padara šausmu spēli biedējošu.

Šis kvalifikācijas darba mērķis ir izveidot biedējošu puzles spēli, ar vairākām un dažādām funkcijām, ar ļoti baismīgu un neērtu atmosfēru, un labu spēlēšanas pieredzi, ko katrs šausmas žanra fanāts varētu izbaudīt.

Pašlaik pastāv šādi analogi:

Inside the Backrooms - šo spēli var spēlēt vairāki cilvēki kopā. Spēles mērķis ir izmeklēt visu mapi, izvairīties no entītijām, un izdzīvot. Spēlei ir vidējas grafikas un vairākas entītijas, bet tikai divas mapes.

The Complex: Found Footage – šajā spēlē ir ļoti kvalitatīvas grafikas, labs skaits dažādu spēlējamu lokāciju un ļoti labi izveidota biedējoša atmosfēra, kas ir visgalvenākais elements šāda žanra šausmu spēlei. Izstrādātājs ir pievienojis VHS kamera efektu, kas palīdz spēlētājam iegremdēties neērtajā atmosfērā. Šī spēle galvenokārt koncentrējās uz lokācijām un to izpētīšanu, tāpēc šeit nav nevienu entītīju ar mērķi nobiedēt spēlētāju.

Ar ko šī lietotne atzšķirsies:

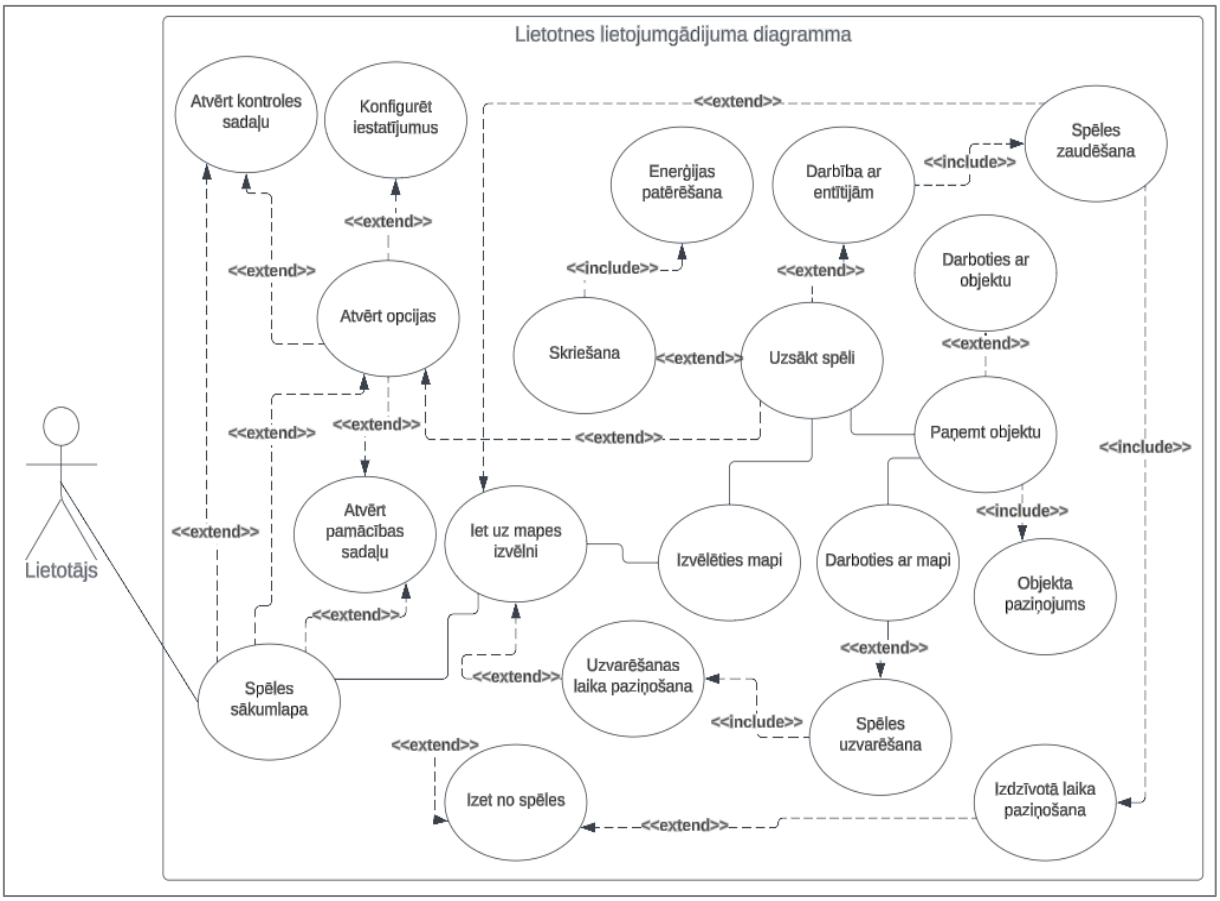
Uncanny Spaces, būs laba un baismīga atmosfēra, labi skaņas efekti, dažādu funkciju un objektu, kas būs unikāli katrai mape, enerģijas sistēma, un katrai mapei savas unikālās entītijas, ar kurām spēlētājs var tikt galā savā veidā, dodot spēlētājam baismīgu un aizraujošu šausmu spēles pieredzi, neprasot stipru datorierīci.

1. UZDEVUMA NOSTĀDNE

Kvalifikācijas darba uzdevums ir izveidot pirmās personas 3D puzles šausmu spēli, kur spēlētājam ir jācenšas izvairīties no entītijām, kas uzbrūk spēlētājam, kamer spēlētājs meklē nepieciešamos priekšmetus lai tiktu līdz izejai un uzvarētu spēli. Spēlei jābūt lietotāja saskarnei, ar kuru var uzsākt spēli izvēlētajā mapē, apskatīt spēles pamācību, spēles kontroles, iziet no spēles, un pielāgot spēles iestatījumus, piemēram kā, skatīšanās ātrums (mouse sensitivity), spēles gaišumu un skaļumu. Katrai mapei jābūt savas unikālās entītijas, un priekšmeti, kurus spēlētājam jāatrod, ar mapes objektiem, ar kuriem spēlētājs var darboties, kā arī spēles atmosfērai un entītijām jābut baismīgām, lai tā skaitītos par šausmu spēli, apgrūtinātu spēles uzvarēšanu, un uzlabotu spēles pieredzi. Pēc spēles beigšanas, spēlētājam jātiek paziņots, cik ilgi gāja spēle. (skat. 1.att.).

Programmai ir jāizpilda šādas funkcijas:

- Mapes izvēlne;
- Kontroles apskatīšanas;
- Spēles pamācības apskatīšanas;
- Spēles nopauzēšana;
- Opcijas izvēlne;
- Pelītes skatīšanās ātruma pielāgošana;
- Spēles gaišuma pielāgošana;
- Spēles skaļuma pielāgošana;
- Spēles uzsākšana;
- Spēlētāja kustība un skatīšanās;
- Enerģijas sistēma, kas ļauj spēlētājam skriet;
- Objektu paņemšana;
- Spēlētāja darbības ar lokāciju;
- Entītiju kustība;
- Spēlētāja darbības ar entītijām;
- Nospēlētā ilguma paziņošana;



1.att. Lietojumgadījuma diagramma

2. PRASĪBU SPECIFIKĀCIJA

2.1. Ieejas un izejas informācijas apraksts

2.1.1. Ieejas informācijas apraksts

Sistēmā tiks nodrošināta šādas ieejas informācijas apstrāde.

1. Informācija par **lietotāju** sastāvēs no šādiem datiem.

- Skatīšanās – spēlētāja kameras virziena kustība – kustinot peli, spēlētāja kamera mainīs virzienu skatoties ekrānā.
- Iešana – spēlētāja kustība uz visām pusēm – nospiežot taustiņus W, A, S, D, spēlētājs kustēsies noteiktajā virzienā. (Piem. W lai ietu uz priekšu)
- Skriešana – spēlētāja noteiktais skriešanas ātrums – nospiežot *Shift* spēlētājs sāk skriet, paātrinot savu kustības ātrumu. (Piem. No 20 uz 50)
- Iestatījumi - spēlētāja izvēlētie iestatījumi – ar pelīti var iestatīt valūtu uz slīdņa, kas ir atbildīgi par dažādiem spēles iestatījumiem. (Piem. No 0 līdz 20).

2. Informācija par **mapēm** sastāvēs no šādiem datiem.

- Mapes izvēle – lietotāja interfeisā, var izvēlēties mapi – spēlētājs var izvēlēties kādā mape viņš spēlēs. (Piem. Noklikšķināt uz 1. mapes)
- Objekta ņemšana – spēlētājs var darboties ar objektiem – spēlētājs var apskatīties uz kādu objektu, un to paņemt. (Piem. Paņemt atslēgu nospiežot E)
- Darbība ar mapi – spēlētājs var darboties ar mapi – spēlētājs var apskatīties uz kādu objektu mapē, un ar to darboties. (Piem. Atvērt durvis nospiežot E)

2.1.2. Izejas informācijas apraksts

1. **Enerģija –** spēlētājam skrienot tērējās enerģija, un tās skaits tiek vizuāli rādīts uz ekrāna. (Piem. Ekrāna apakšmala vidū taisnstūris, ar enerģijas valūtu baltā krāsā uz pelēka fona.)
2. **Ekrāna kustība** - spēlētājam staigājot vai skrienot, atbilstoši kustēsies ekrāns.(Piem. Staigājot lēnam kustās ekrānas, bet skrienot - ātri kustās ekrāns.)
3. **Soļu skaņas** - spēlētājam staigājot vai skrienot, atbilstoši tiks izdvests soļu skaņas. (Piem. Staigājot tiks izdvesti staigāšanas audio, bet skrienot – skriešanas audio.)
4. **Darbības ar priekšmetu** - apskatoties uz kādu objektu, spēlētājam pateiks, kā ar to var darboties. (Piem. “Press E to pick up key.”)
5. **Inventārs** - kad spēlētājs ir paņēmis kādu priekšmetu, tas priekšmets uzrādīsies ekrāna malā, lai paziņotu spēlētājam, ka priekšmets atrodas inventārā. (Piem. Atslēgas ikona ekrāna kreisajā apakšējā)
6. **Laiks** - ekrāna augšmalā pa vidu atrodas laiks, kurš rāda, cik ilgi iet spēlē. (Piem. ”07:10”)
7. **Izspēlētais laiks** - kad spēlētājs ir, vai nu uzvarējis spēli, vai to zaudējis, tiks parādīts cik laiks pagāja spēles laikā. (Piem. “You survived for 7 minutes and 10 seconds.”)
8. **Līmeņa iziešana** - veiksmīgi izbēgot no mapes jeb izpildot līmeni, spēlētājam paziņos, ka viņš ir izbēdzis. (Piem. “You have successfully escaped!”)
9. **Zaudēšana** - ja spēlētāju noķer kāda entītija, izleks izbīlis, un tiks paziņots, ka esat zaudējis. (Piem. “You died.”)

2.2. Funkcionālās prasības

1. Jānodrošina iespēja lietotājam pārvietoties lietotāja interfeisa.
    1.1. Jābūt korektai interfeisa navigācijai.
    1.2. Noklikšķinot *Exit*, jāiziet no spēles.
    1.3. Noklikšķinot *Play*, jāaiziet uz mapes izvēlni.
    1.4. Noklikšķinot *Controls*, jāaiziet uz vadības lapu.
    1.5. Noklikšķinot *How to play*, jāaiziet uz pamācības lapu.
    1.6. Noklikšķinot kādu no mapēm, jāuzsāk spēle noteiktajā mapē.
    1.7. Zaudēšanas ekrānā, noklikšķinot uz R aiziet uz sākumlapu.
    1.8. Uzvarēšanas ekrānā, noklikšķinot Select Map aizvest uz mapes izvēlni.
2. Jānodrošina lietotāja darbības ar opcijām.
    2.1. Jābūt korektai opciju izvēlnes navigācijai.
    2.2. Nospiežot taustiņu "Q" jāatveras opcijas izvēlnei.
    2.3. Nospiežot taustiņu "Q", kamēr opcijas ir atvērtas, tām jāaizveras.
    2.4. Nospiežot Resume jāatsāk spēli un jāaizver opcijas.
    2.5. Nospiežot Settings jāatver iestatījumus.
    2.6. Nospiežot Controls jāatver vadības lapu.
    2.7. Nospiežot Main Menu jāaiziet uz galveno izvēlni.
    2.8. Konfigurējot Sensitivity (pelītes ātrums), jāmainas to valūtai.
    2.9. Konfigurējot Brightness (spēles gaišums), jāmainas to valūtai.
    2.10. Konfigurējot Volume (spēles skaļums), jāmainas to valūtai.
3. Jānodrošina spēlētāja kustību un enerģijas darbošanos.
    3.1. Spēlētājam jāsāk kustēties nospiežot taustiņus W, A, S, D.
    3.2. Spēlētājam jāspēj skatīties apkārt ar datorpeles kustību.
    3.3. Nospiežot Shift kamēr spēlētājs kustās, sākt skriešanu.
    3.4. Radīt soļu skaņu, kad spēlētājs staigā.
    3.5. Radīt skriešanas skaņu, kad spēlētājs skrien.
    3.6. Kustināt ekrānu atbilstoši staigāšanas soļiem.
    3.7. Kustināt ekrānu atbilstoši skriešanas soļiem.
    3.8. Patērēt enerģiju, kad spēlētājs skrien.
    3.9. Gūt atpakaļ enerģiju, kamēr spēlētājs neskrien.
    3.10. Samazināt skriešanas ātrumu, ja enerģija ir zema.
4. Jānodrošina lietotāja darbības ar mapi un objektiem.

4.1. Apskatoties uz objektu, ar kuru var darboties, parādīt paziņojumu.

4.2. Apskatoties uz objektu, ar kuru var darboties tikai tad, ja ir vajadzīgais priekšmets, paradīt paziņojumu.

4.3. Nospiežot izvēlēto taustiņu, spēlētājs veic darbību ar objektu un izrada skaņu.

4.4. Spēlētājam skaroties ar izeju, aizvest uz uzvarēšanas ekrānu un parādīt ziņojumu.

4.5. Spēlētājam nospiežot taustiņu E uz izejas, aizvest uz uzvarēšanas ekrānu un parādīt ziņojumu.

5. Jānodrošina lietotāja darbību ar entītijām.

5.1. Entītijai radīt skaņu no sevis, kas palielinās tuvojoties spēlētājam.

5.2. Spēlētājam saskaroties ar entītīju, radīt skaņu, parādīt ziņojumu un aizvest uz zaudēšanas ekrānu.

5.3. Entītijai apstāties, kad spēlētājs uz to skatās.

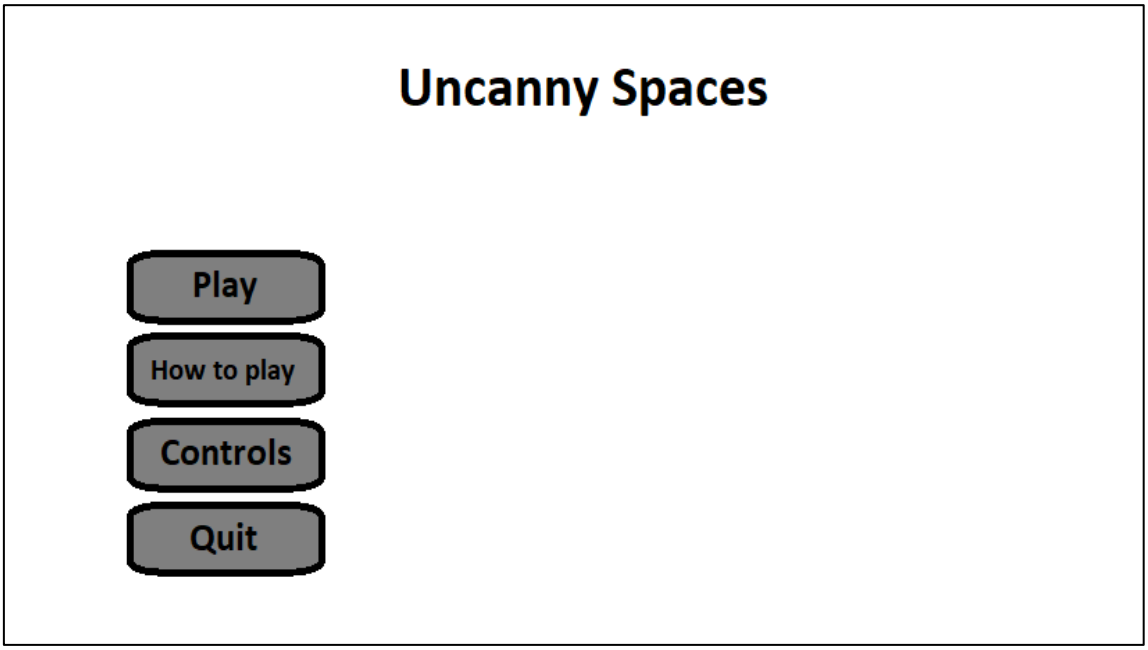
5.4. Entītiju parādīšanos mapē, pēc priekšmeta paņemšanas.

5.5. Entītiju lokācijas maiņu, kad uz to spēlētājs skatās pietiekami ilgi.

2.3. Nefunkcionālās prasības

1. Sistēmas saskarnes valodai ir jābūt angļu valodai.
2. Jānodrošina spēles dinamisku pielāgošanos ekrāna izmēriem.
3. Interfeisam ir jābūt izskatīgam un viegli lietojamam.
4. Tekstam ir jābūt viegli izlasāmam.
5. Dizainam jābūt piemērotam šausmu spēlei.

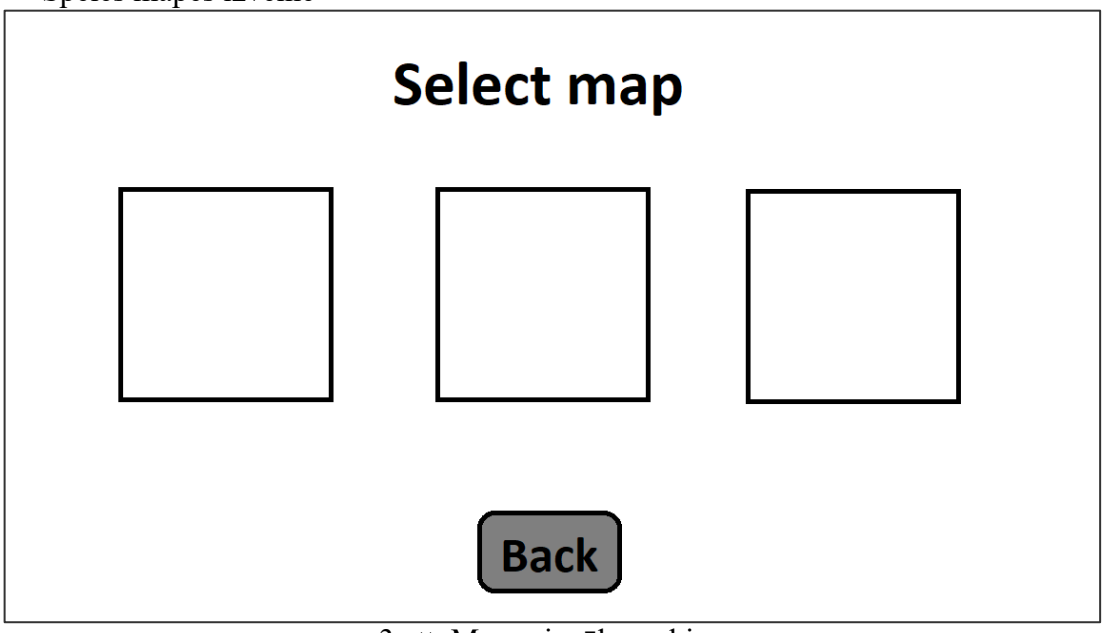
- Spēles sākumlapa



2.att. Sākumlapas skice

Skice attēlo galveno lapu, spiežot *Play* aizved uz mapes izvēlni, spiežot *Exit,* lietotājs iziet no spēles.

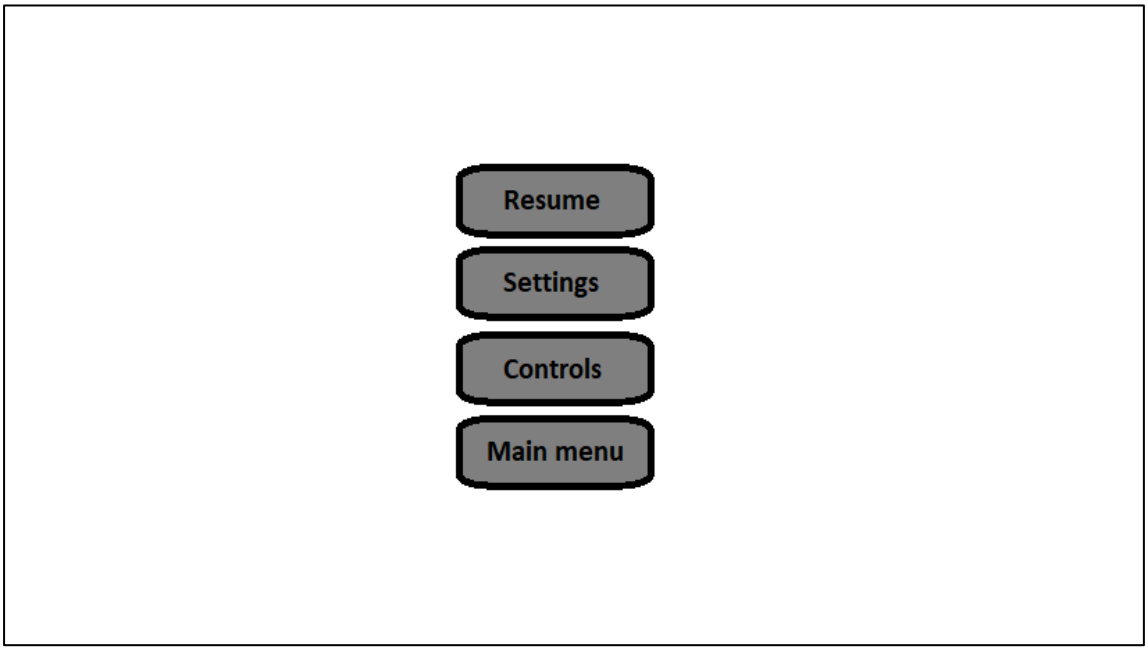
- Spēles mapes izvēlne



3.att. Mapes izvēlnes skice

Šajā skicē ir redzama izvēlne no 3 mapēm, kuras var spēlēt, uz tām noklikšķinot.

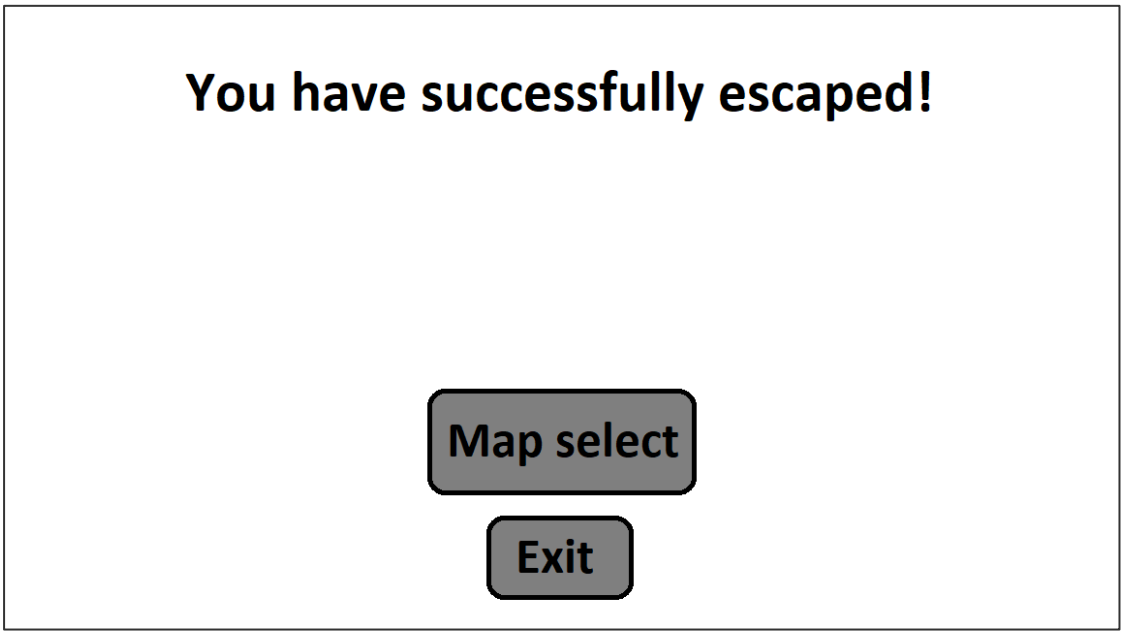
- Opcijas izvēlne



4.att. Opcijas izvēlnes skice

Šī skice attēlo opcijas izvēlni, kuru spēlētājs var atvērt spēles laikā.

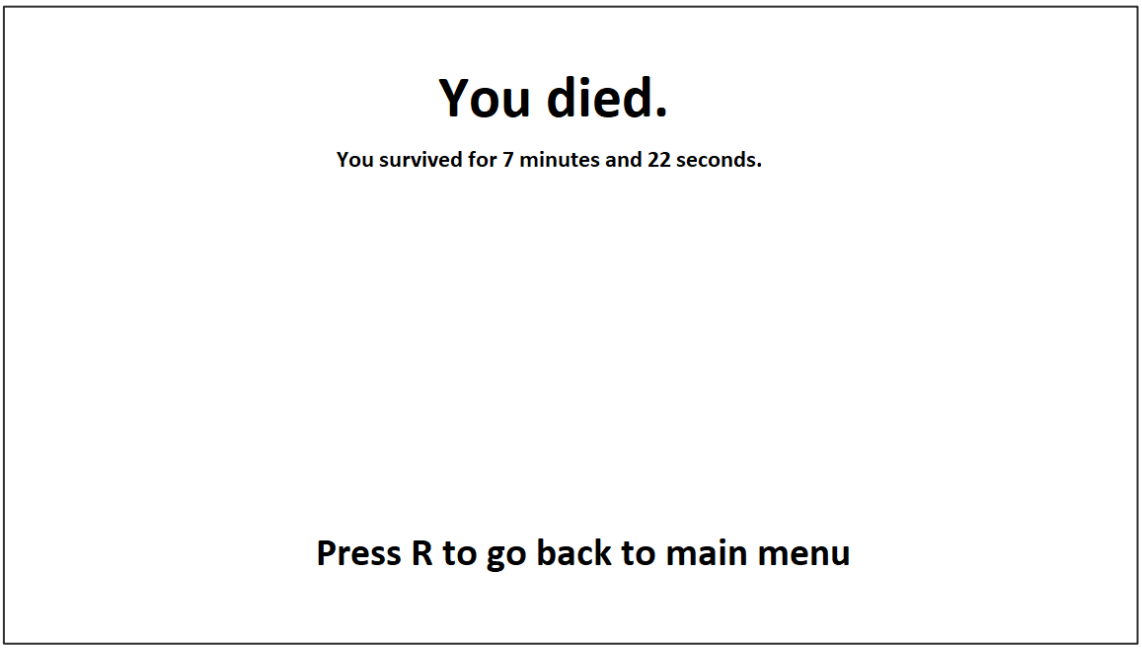
- Uzvarēšanas ekrāns



5.att. Uzvarēšanas ekrāna skice

Ši skīce attēlo uzvarēšanas ekrānu, kurā spēlētājs var nokļūt izbēgot no mapes.

- Zaudēšanas ekrāns



6.att. Zaudēšanas ekrāna skice

Ši skīce attēlo zaudēšanas ekrānu, kur spēlētājs nokļūst saskaroties ar entītiju.

3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS

Lai realizētu projektu, tika izmantots *2020.3.33f1* versijas *Unity,* kas ir starp-platformu spēļu dzinis un programmēšanas vide spēlēm. *Unity* izmanto DirectX11 grafisko tehnoloģiju, kas uzlabo spēles stabilitāti, un NET 4.x struktūru, kas nodrošina programmas darbību uz vairākām, un dažādām aparatūrām.

Es nolēmu lietot 2020. gada versiju, tāpēc ka pielietojot jaunākās versijas var rasties nesaderības kļūdas starp kodu un *Unity* paketēm, un novērojamu uzlabojumu, kas man būtu lietderīgi, jaunākajai versijai nebija.

Koda rakstīšanas programmēšanas valoda *Unity* ir C#. Visas funkcijas un metodes, kas nodrošina spēles darbību, tika rakstītās palīgprogrammā *Microsoft Visual Studio*. Lai nodrošinātu lietotāja interfeisa, tika importērta pakete *Unity UI*.

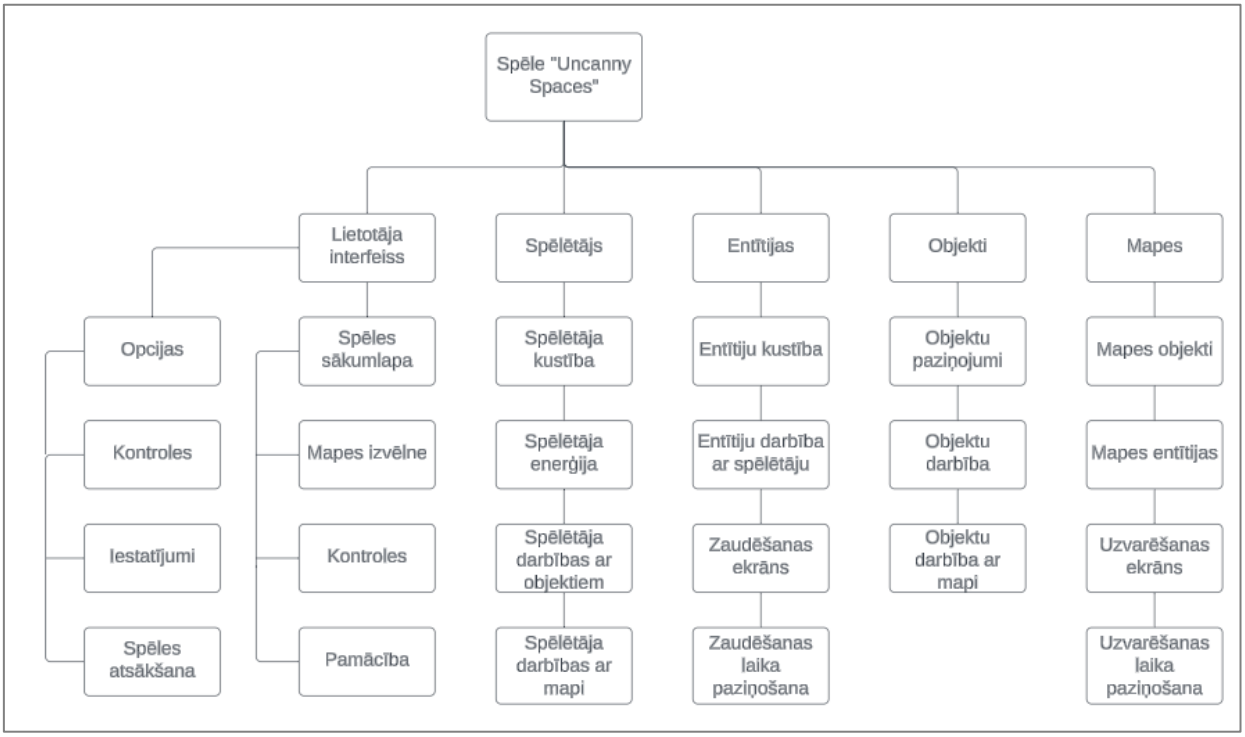
Spēlē izmantotie audio faili tikai ņemti no tīmekļa vietnēm *epidemicsound.com* un *pixabay.com.* Tekstūras tika meklētas *google.com,* un visi objekti tika izveidoti lietojot standarta *Unity* figūras.

4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

4.1. Sistēmas strukturālais modelis

4.1.1. Sistēmas arhitektūra

**Tīmekļa foruma funkcionālā dekompozīcijas diagramma (skat. 7.att.) sastāv no 3 apasskistēmam.

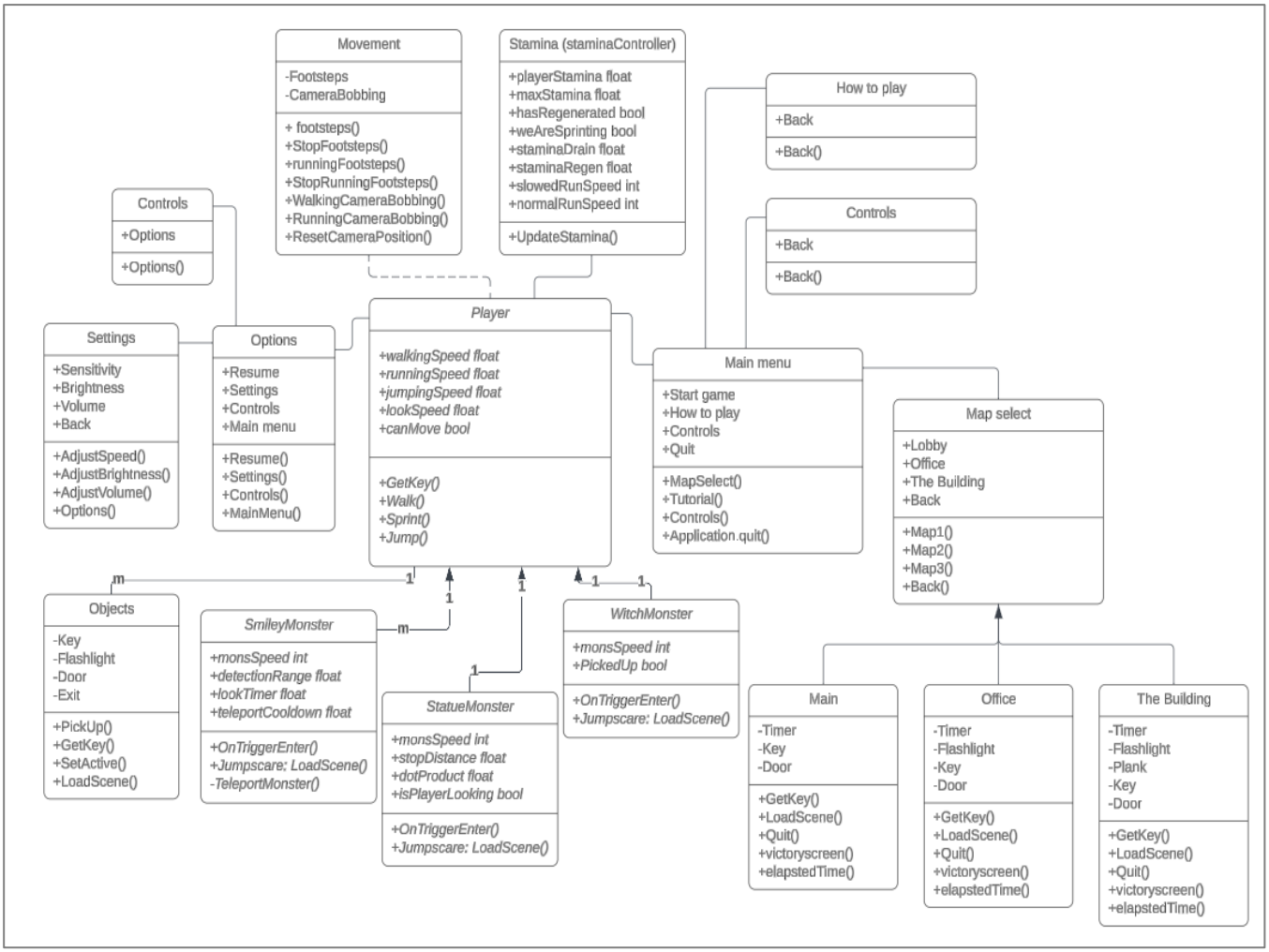


7.att. Funkcionālās dekompozīcijas diagramma

1. **Lietotāju interfeiss** – lietotājs var vadīties pār galveno izvēlni, izvēloties mapi, apskatīt kontroles un spēles pamācibu, un uzsākt spēli.
   1.1. Opcijas – atverot opcijas izvēlni, lietotājs var iet uz kontroles sadaļu, iestatījumu sadaļu, un atsākt spēli.
2. **Spēlētājs** – lietotājs uzsākot spēli, var skatīties, staigāt, lekt un skriet apkārt pa mapi 3 dimensijās, spēlētājam skrienot patērējās enerģija. Spēlētājs var darboties ar objektiem un tos paņemt, kā arī darboties ar mapi.
3. **Entītijas** – entītijas kustās pa mapi, sekojot spēlētājam, ja kāda entītija skarās spēlētājam, tad spēlētājs tiek nobiedēts, spēle beidzas, un tiek paziņots nospēlētais ilgums.

4. **Objekti** – spēlētājam skatoties uz objektiem, tiek rādīts informējošs paziņojums, spēlētājam darbojoties ar objektu, tiek izradīta skaņa, un atslēdzas jaunas darbības ar mapi.
5. **Mapes** – katrā mapē ir savi noteiktie objekti, entītijas, un izeju, ar ko sastopoties spēlētājs uzvar spēli, kā arī tiek pazinots ilgums, cik ilgi tika nospēlēts.

4.1.2. UML klašu diagramma



8.att. UML klašu diagramma

- "Player" ir saistīts ar "Movement" ar tiešo pārtraukto līniju, jo tā ir
- "Player" ir saistīts ar "Main menu" ar tiešo saiti, jo tā ir lietotāja saskarne, ta ir 3 apakšgrupas, "How to play", "Controls", "Map select".
- "Map select" ir 3 apakšgrupas, "Main", "Office", "The Building", tās ir visas spēlējamās mapes.
- "Player" ir saistīts ar "Options" ar tiešo līniju, jo tā ir lietotāju saskarne, tai ir apakšsadaļa "Options".

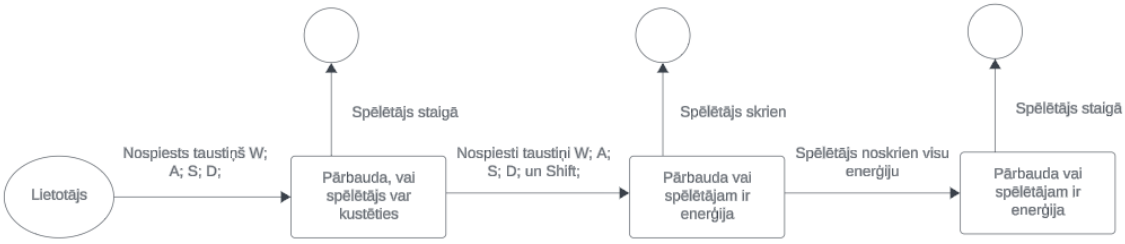
- “Options” ir divas sadaļas “Controls” un “Settings”, Controls var apskatīt spēles kontroles, Settings var regulēt iestatījumus.
- “Player” ir saistīts ar “Objects” ar viens pret daudziem, jo spēlētājs var darboties ar vairākiem priekšmetiem.
- “Player” ir saistīts ar entītiju “SmileyMonster” ar viens pret daudziem, jo pastāv vairāku tādu entītiju.
- “Player” ir saistīts ar “WitchMonster” ar viens pret vienu, jo ir tikai viena šāda entītija.
- “Player” ir saistīts ar “StatueMonster” ar viens pret vienu, jo ir tikai viena šāda entītija.

Spēlētājs lietojot lietotāja interfeisu, var izvēlēties, kuru no vairākām mapēm viņš vēlās spēlēt, un uzsākt spēli. Katrā mapē ir viens vai vairāki objekti, ar ko spēlētājs var darboties. Spēlētājs ir tikai viens, bet vienā mapē var būt vairākas entītijas, kas darbojas ar spēletāju. Spēlētājam ir enerģija, kas tiek patērēta kad viņš skrien, un kā arī animācijas priekš soļiem, kad spēlētājs kustās vai skrien.

4.2. Sistēmas funckionālais modelis

4.2.1. Datu plūsmu modelis

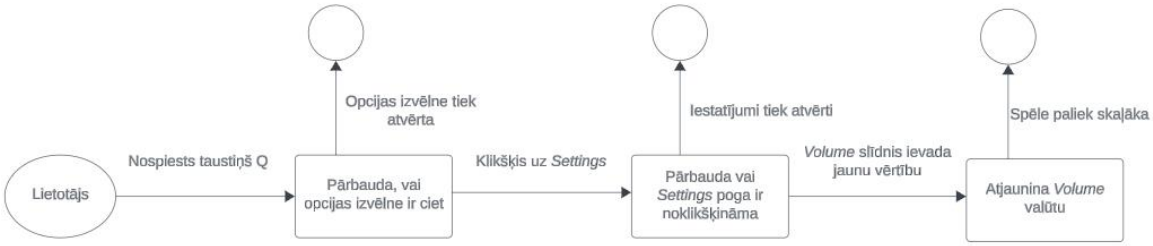
1. Spēlētājs iztēre visu enerģiju (skat. 9.att.).



9.att. Enerģijas iztērēšanas datu plūsmas modelis

Lietotājs nospiež kādu no kustības taustiņiem (W, A, S, D), spēle pārbauda vai spēlētājs var kustēties, spēlētājs sāk kustību. Pēctam spēlētājs nospiež *Shift* kopā ar kustības taustiņiem, spēle pārbauda vai enerģijas valūta ir virs 0, un spēlētājs sāk skriet. Spēlētājs skrien līdz enerģija noiet līdz 0, spēle atkārtoti pārbauda vai ir enerģija, un kad tā beidzas, spēlētājs var tikai staigāt.

1. Lietotājs caur iestatījumiem palielina spēles skaņu (skat. 10.att.).



10.att. Spēles skaņas palielināšanas datu plūsmas modelis

Lietotājs nospiež taustiņu Q, spēle pārbauda vai opcijas izvēlne ir aizvērta, opcijas izvēlne atvēras, lietotājs noklikšķina uz *Settings*, tiek pārbaudits vai uz *Settings* var noklikšķināt, atverās iestatījumi. Lietotājs uz *Volume* slīdna iestata jaunu valūtu ar savu pelīti, *Volume* valūta tiek atjaunināta, un spēle paliek skaļāka.

1. Spēlētājs atslēdz durvis (skat. 11.att.).

11.att. Durvs atvēršanas datu plūsmas modelis

Lietotājs nospiež taustiņu E, spēle pārbauda, vai spēlētājs skatās uz atslēgām, spēlētājs paņem atslēgas, lietotājs nospiež taustiņu E, spēle pārbauda vai spēlētājs skatās uz durvīm, durvis tiek atvērtas.

5. DATU STRUKTŪRU APRAKSTS

Šajā sistēmā nepastāv datu bāzes, tāpēc tika aprakstīti visi sistēmas izmantojamie mainīgie, to funkcijas, un kā tie darbojas savā starpā. 5.1. attēlā ir redzama saišu shēma. Sistēma sastāv no 8 grupām ar datiem:

- **"Spēlētājs"** datu tabulā atrodas visi mainīgie, kas ir saistīti ar spēlētāja kontroli un "**Enerģija"** datu tabulā ir visi mainīgie, kas atbild par enerģijas funkciju, un kā tā ietekmē spēlētāja kustību.
- **"Entītijas"** datu tabulā ir viss, par entītiju kustību, lokāciju, un darbībām ar spēlētāju.
- **"Laiks"** tabulā ir visi mainīgie, kas nodrošina spēles laika skaitīšanu, to paziņošanu, un animāciju.
- **"Kustība"** dati ir atbildīgi gan par spēlētāja kustības animācijām, gan audio.
- **"Priekšmeti"** datu tabulā ir mainīgie, kas ir atbildīgi par priekšmetu pareizu funkciju, un "to paziņojumu parādīšanu.
- **"Opcijas"** tabulā ir visi mainīgie, kas ir atbildīgi par spēles iestatījumiem.
- **"Lokācijas"** datu tabulā ir mainīgie, kas atbild par lokācijas objektu stāvokli un scēnu, vai mapes nosaukumu.

1. tabula

"Spēlētājs" datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	walkingSpeed	float	2	Spēlētāja iešanas ātrums
2.	runningSpeed	float	2	Spēlētāja skriešanas ātrums
3.	jumpSpeed	float	2	Spēlētāja lekšanas ātrums
4.	lookSpeed	float	2	Spēlētāja skatīšanās ātrums
5.	lookXLimit	float	2	Spēlētāja skatīšanās vertikālais limits
6.	gravity	float	2	Spēlētāja gravitāte
7.	canMove	bool	-	Pārbauda vai spēlētājs var kustēties

Spēlētājs var darboties ar visiem priekšmetiem, entītijām un lokācijām.

2. tabula

“Enerģija” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	playerStamina	float	2	Spēlētāja enerģija
2.	maxStamina	float	2	Spēlētāja maksimālā enerģija
3.	hasRegenerated	bool	-	Pārbauda vai enerģija ir pilna
4.	weAreSprinting	bool	-	Pārbauda vai spēlētājs skrien
5.	staminaDrain	float	2	Enerģijas patērēšanas ātrums
6.	staminaRegen	float	2	Enerģījas atjaunošanas ātrums
7.	slowedRunSpeed	int	2	Palēlinātās skriešanas ātrums
8.	normalRunSpeed	int	2	Normālās skriešanas ātrums

Šī tabula darbojas tieši ar spēlētāju.

3. tabula

“Entītijas” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	monsSpeed	int	2	Entītiju ātrums
2.	detectionRange	float	2	Entītiju atklāšanas distance
3.	isPlayerLooking	bool	-	Pārbauda vai spēlētājs skatās uz entītiju
4.	lookTimer	float	2	Skaita cik ilgi spēlētājs skatās uz entītiju
5.	teleportCooldown	float	2	Ilgums, cik ilgi spēlētājam jāskatās uz entītiju
6.	stopDistance	float	3	Distance, kurā entītija apstājas, ja uz to skatās spēlētājs
7.	distanceToPlayer	float	3	Distance starp spēlētāju un entītiju
8.	dotProduct	float	3	Pārbauda vai entītija ir redzama spēlētājam

Entītijas var darboties ar spēlētāju (uzbrukt spēlētājam), priekšmetiem (parādīties mapē pēc priekšmetu paņemšanas), un mainīt scēnu (iet uz zaudēšanas ekrānu), ja spēlētājs tiek noķerts.

4. tabula

“Laiks” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	elapsedTime	float	30	Nospēlētais ilgums
2.	minutes	int	2	Minūtes skaitītājs
3.	seconds	int	2	Sekunžu skaitītājs
4.	minutesText	string	30	Minūšu teksts
5.	secondsText	string	30	Sekunžu teksts
6.	beepInterval	float	1	Kameras efekta sarkanā punkta pīkstēšanas intervāls

Ar šiem datiem darbojas spēlētājs (kamēr spēle iet), un lokācija (kad spēle beidzas), paziņojot elapsedTime.

5. tabula

“Kustība” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	isFootstepActive	bool	-	Pārbauda vai spēlētājs staigā
2.	isRunningFootstepActive	bool	-	Pārbauda vai spēlētājs skrien
3.	walkingBobbingSpeed	float	2	Iešanas kameras kustības ātrums
4.	walkingBobbingAmount	float	2	Iešanas kameras kustības daudzums
5.	runningBobbingSpeed	float	2	Skriešanas kameras kustības ātrums
6.	runningBobbingAmount	float	2	Skriešanas kameras kustības daudzums
7.	walkingKeyPressed	bool	-	Pārbauda vai kāds no iešanas taustiņiem ir nospiests
8.	shiftKeyPressed	bool	-	Pārbauda vai skriešanas (Shift) taustiņš ir nospiests

Šie dati ir tieši saistīti ar spēlētāju.

6. tabula

“Priekšmeti” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	keyfound	bool	-	Pārbauda vai priekšmets ir atrasts
2.	PickedUp	bool	-	Pārbauda vai priekšmets ir paņemts
3.	displayDuration	float	-	Paziņojuma ilgums
4.	fadeDuration	float	-	Paziņojuma izzušanas ilgums

Ar priekšmetiem var darboties spēlētājs (spēlētājs var paņemt priekšmetus). Dati ir saistīti ar spēlētāju, entītijām, un lokācijām.

7. tabula

“Opcijas” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	SensitivityKey	string	30	Spēlētāja skatīšanās ātrums (Mouse sensitivity)
2.	volume	float	3	Spēles skaļums
3.	savedBrightness	float	3	Spēles gaišums

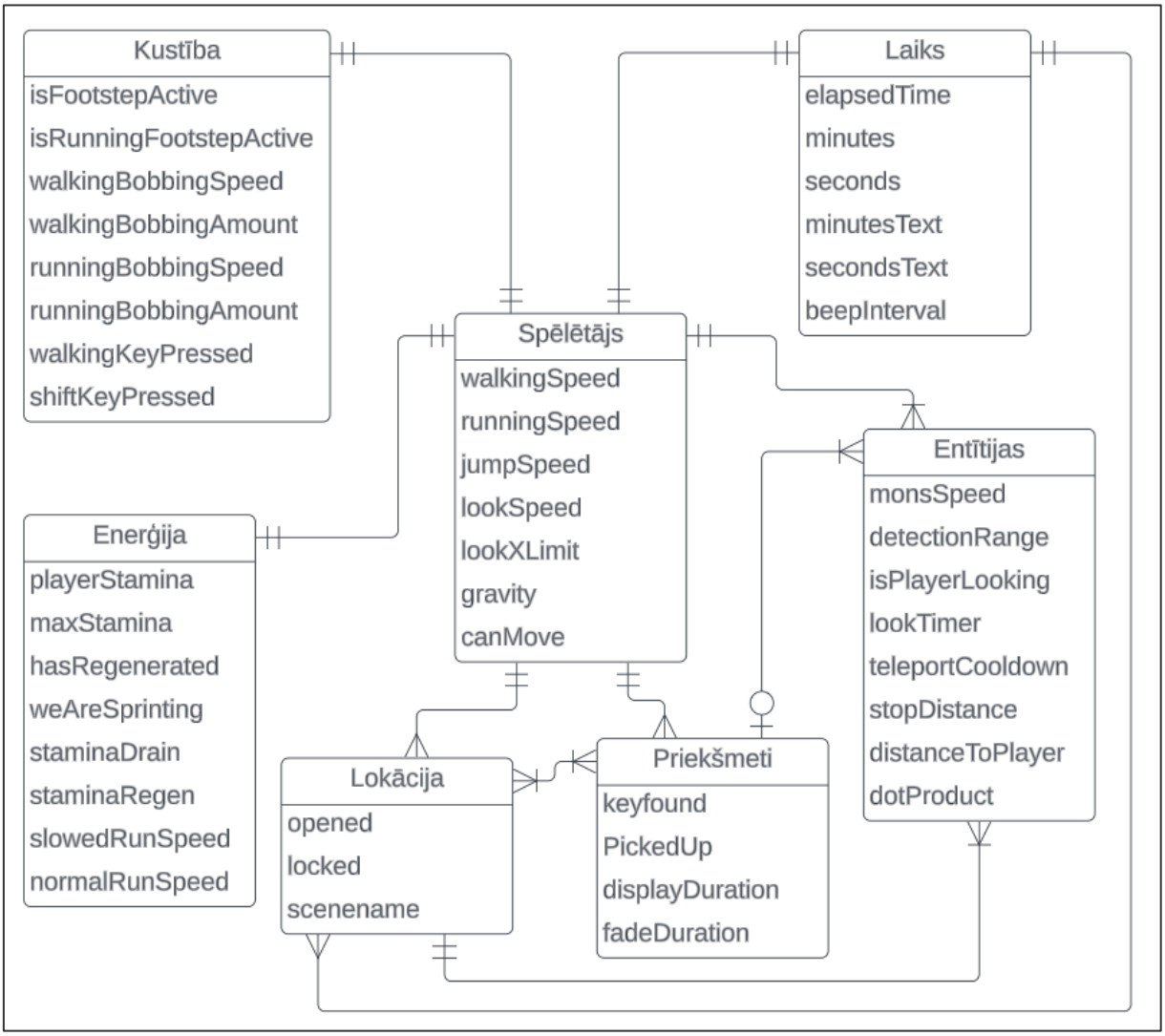
Dati ir tieši saistīti ar spēlētāju.

8. tabula

“Lokācija” datu tabula

Nr	Lauka nosaukums	Tips	Izmērs	Apraksts
1.	opened	bool	-	Pārbauda vai objekts ir atslēgts
2.	locked	bool	-	Pārbauda vai objekts ir aizslēgts
3.	scenename	string	30	Scēnas nosaukums

Lokācijas dati darbojas ar spēlētāju (spēlētājs atrod izeju un iziet mapi) un priekšmetiem (Piem. durvis kļūst atveramas pēc atslēgas paņemšanas).



12.att. Datu tabulu saišu shēma

6. LIETOTĀJA CEĻVEDIS

6.1. Sistēmas prasības

Sistēma tika realizēta kā instalējama aplikācija uz datora ierīcēm. Lai veiksmīgi palaistu aplikāciju, ir jāievēro šādi pieprasījumi:

9. tabula

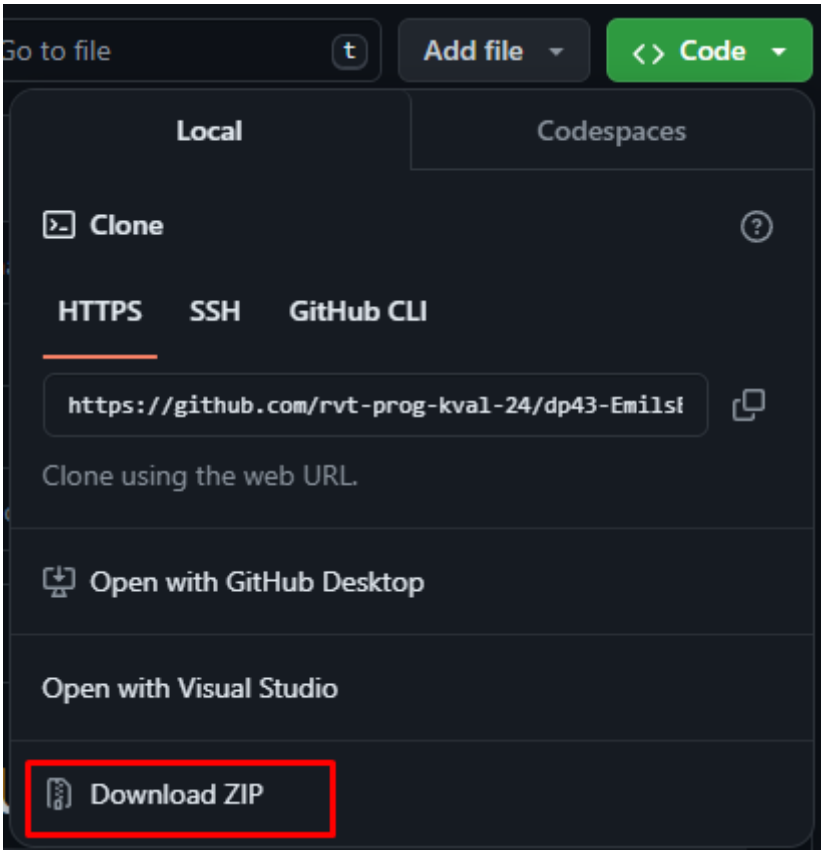
Operētājsistēma	**Windows**	**macOS**	**Linux**
Operētājsistēmas versija	Windows 7, Windows 10, Windows 11	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, and CentOS 7
Datora CPU bit versija	x86, x64	x86, x64	x86, x64
Grafikas API	DX10, DX11, DX12 spējīga	*Metal* spējīga Intel vai AMD grafikas kartes.	*OpenGL* 3.2+, *Vulkan* spējīga

Tiek rekomendēts, ka datora ierīeci ir vismaz 4GB RAM, un 10GB brīvas vietas uz diska.

6.2. Sistēmas instalācija un palaišana

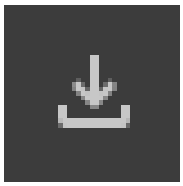
Lai veiksmīgi palaistu aplikāciju, jāseko šiem soļiem:

1. Ielādējiet ZIP failu no GitHub repozitorijas (skat. 11.att.).

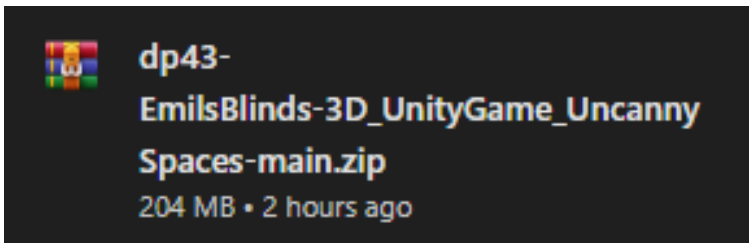


13.att. Instalācijas pirmā soļa bilde

2. Atveriet ielādēto zip failu (skat. 12.att., 13.att.).

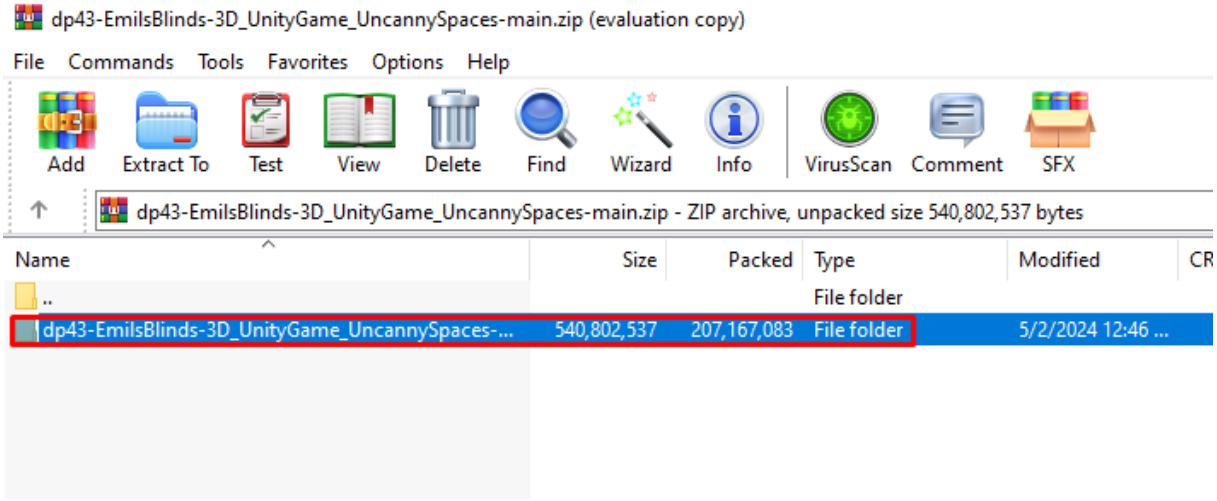


14.att. Instalācijas otrā soļa bilde



15.att. Instalācijas otrā soļa bilde

3. Izvelciet ZIP failu jums vēlamā vietā. (skat. 14.att.,15.att.).



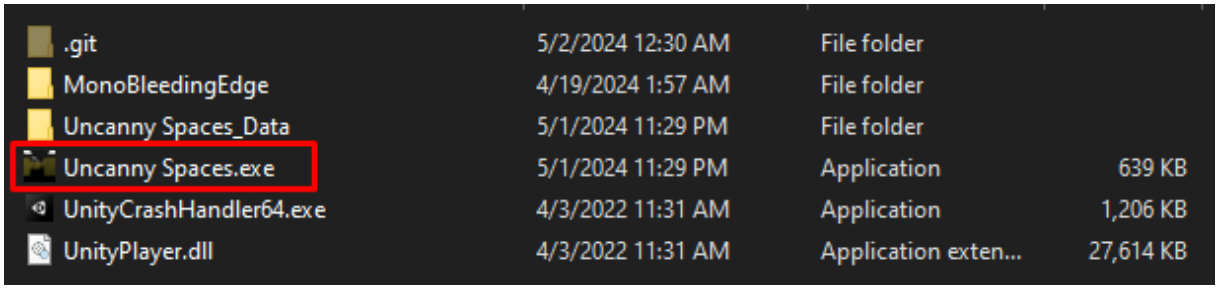
16.att. Instalācijas trešā soļa bilde

17.att. Instalācijas trešā soļa bilde

4. Atverat mapi (skat. 16.att.).

18.att. Instalācijas ceturtā soļa bilde

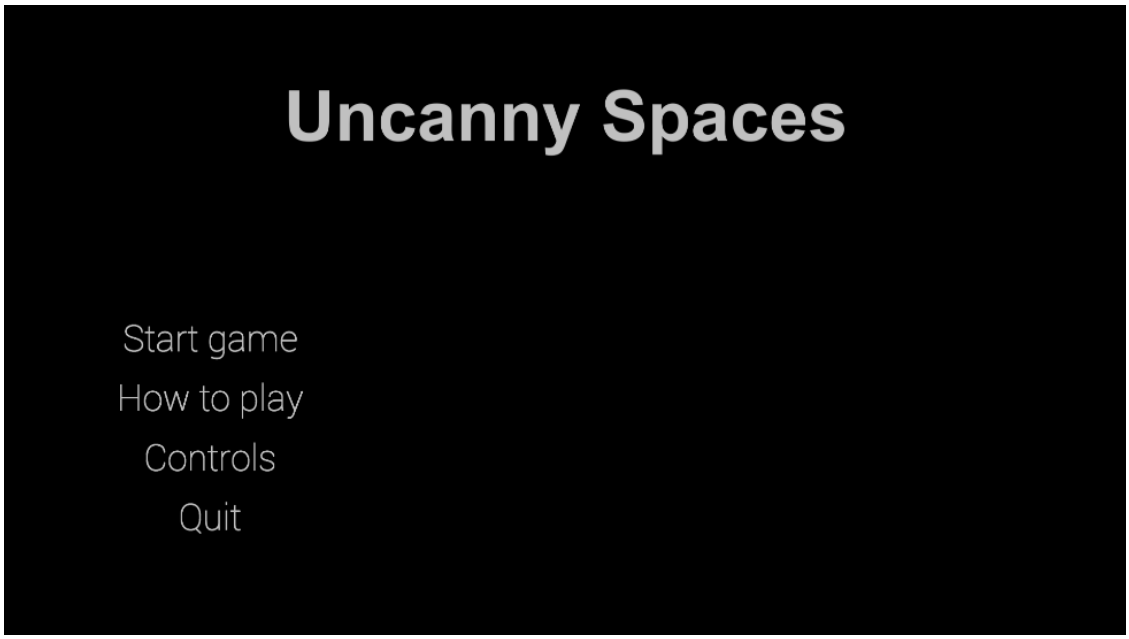
5. Palaižat UncannySpaces.exe (skat. 17.att.).



19.att. Instalācijas piektā soļa bilde

6.3. Programmas apraksts

Atverot spēli, būs redzams Main Menu.

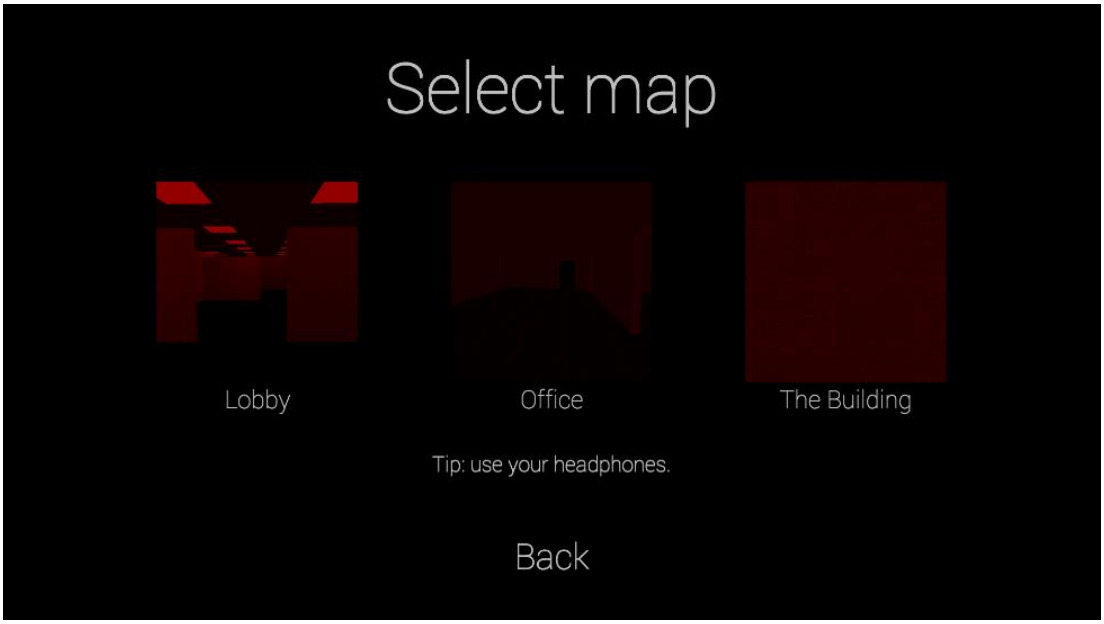


20.att. Main Menu izvēlne

Main Menu ir 4 opcijas:

1. Start game, uz ko noklikšķinot aizvedīs uz mapes izvēlni;
2. How to play, uz ko noklikšķinot aizvedīs uz pamācības lapu;
3. Controls, uz ko noklikšķinot aizvedīs uz kontroles lapu.
4. Quit, uz ko noklikšķinot spēles tiks aizvērta.

Tikta atvērta mapes izvēlne.

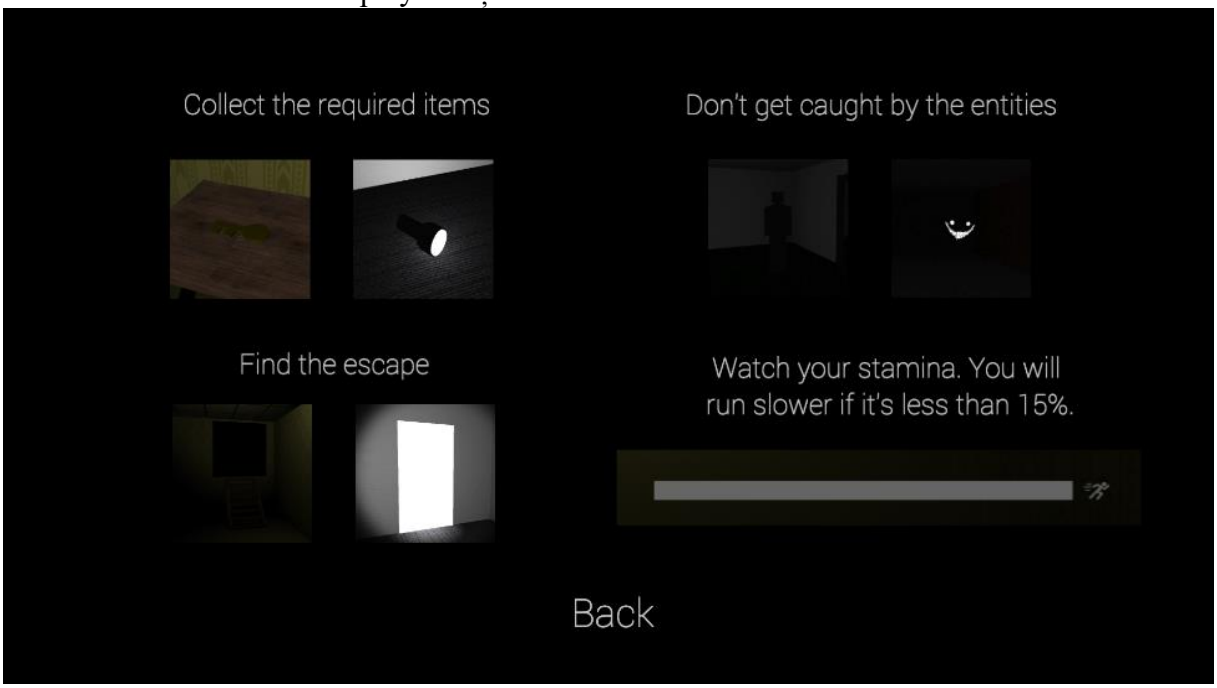


21.att. Mapes izvēlne

Mapes izvēlnē ir 4 opcijas:

1. Lobby, uz ko noklikšķinot spēles sāksies pirmajā mapē (Lobby);
2. Office, uz ko noklikšķinot spēles sāksies otrajā mapē (Office);
3. Building, uz ko noklikšķinot spēles sāksies trešajā mapē (Building).
4. Back, uz ko noklikšķinot ies atpakal uz Main Menu.

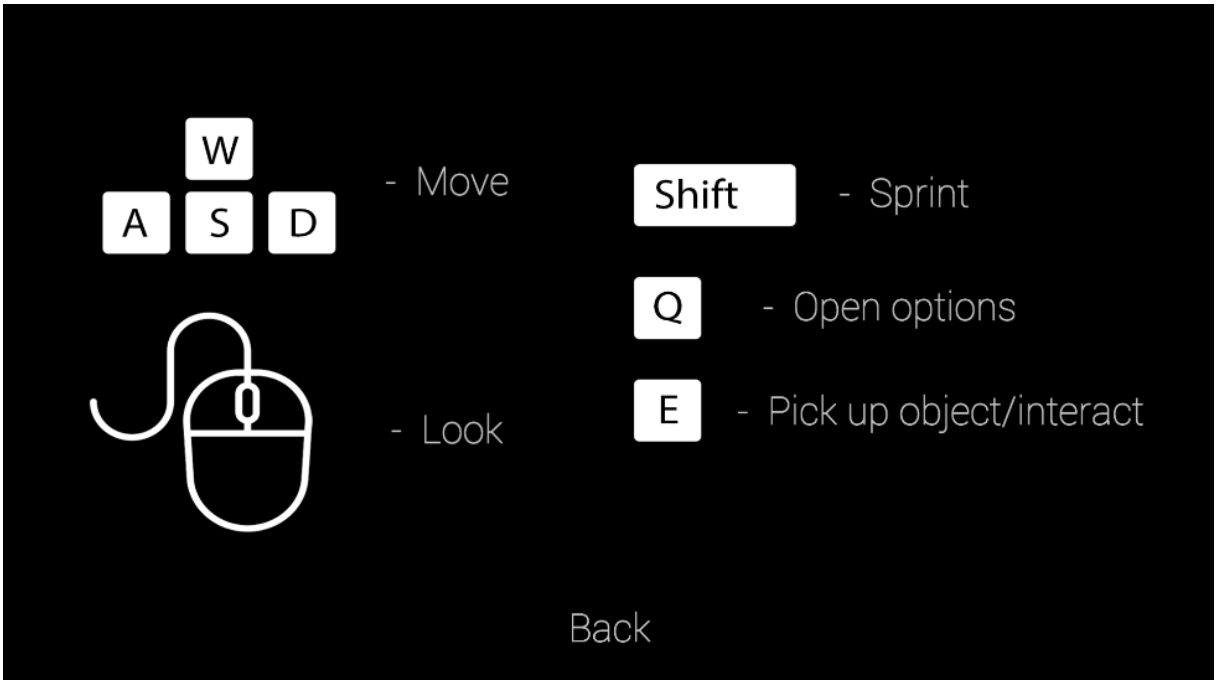
Tikta atvērta How to play sadaļa.



22.att. How to play sadaļa

Šeit var apskatīt spēles pamācību. Noklikšķinot Back, ies atpakal uz Main Menu.

Tiek atvērta Controls sadaļa. (skat. att. .)



23.att. Kontroles sadaļa

Šeit var apskatīt visas kontroles spēlei. Noklikšķinot Back, ies atpakal uz Main Menu.

Tiek uzsākta spēle pirmajā mape Lobby. (skat. att. .)

24.att. Kontroles sadaļa

Spēlētājs var skatīties apkārt ar datorpeli, un kustēties lietot W, A, S, D, ja staigājot nospiež Shift, spēlētājs skries un tērēsies enerģija, kas ir redzama ekrāna apakšvidū.

25.att. Darbība ar objektu

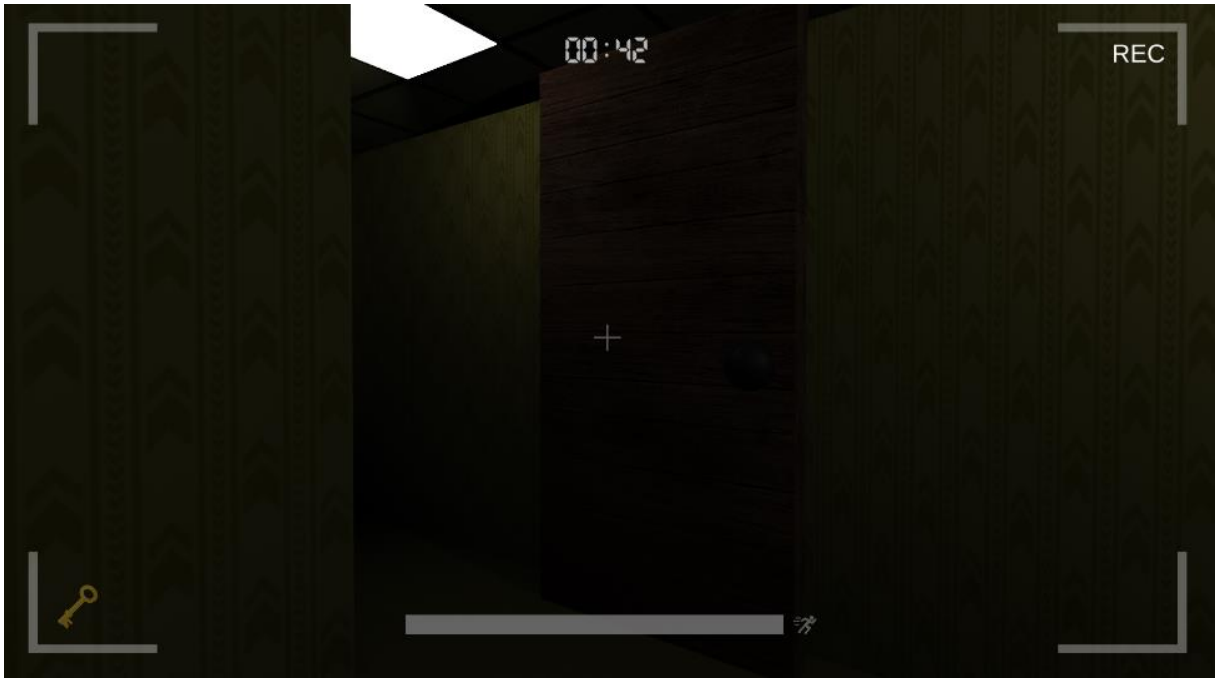
Paskatoties uz objektu, ar kuru var darboties, tiks parādīts paziņojums, kā ar to darboties. (piem. “Nospied E, lai paņemtu”)

26.att. Entītija pirmajā mapē Lobby

Ja objekts tiek paņemts, mapē parādās entītija, kas sekos spēlētājam.

27.att. Iespējama darbība ar durvīm

Ja spēlētājam ir atslēga, tad ir iespējams atvērt durvis ar taustiņu E.

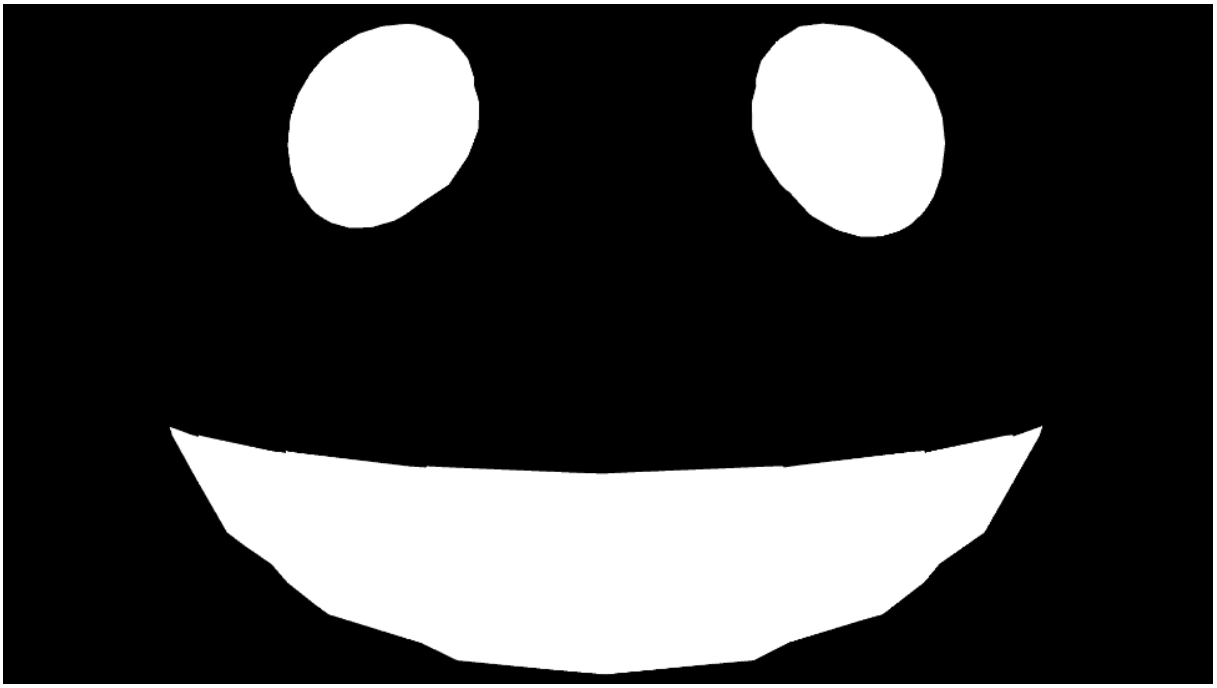


28.att. Atvērtas durvis

Pēc E taustiņa nospiešanas, durvis atverās.

29.att. Izeja pirmajā mapē Lobby

Nospiežot taustiņu E, spēlētājs izbēg no mapes un uzvar spēli.



30.att. Izbīlis

Izbīlis, kas biedē spēlētāju.

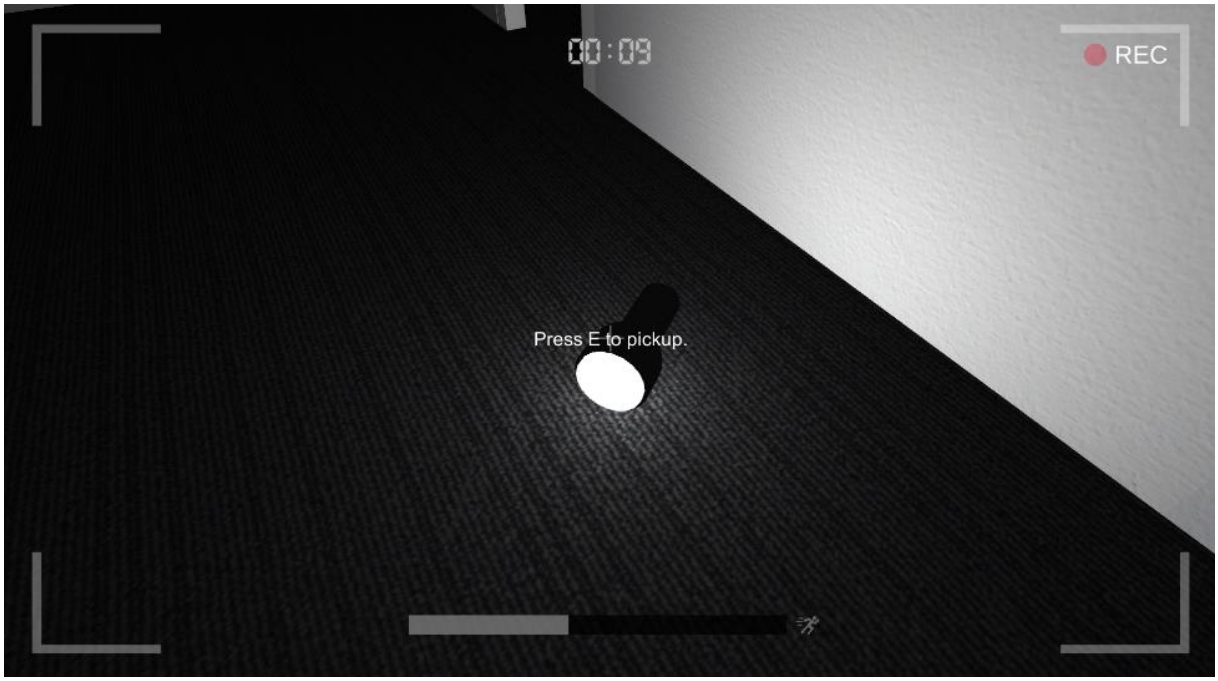
31.att. Zaudēšanas ekrāns

Ja spēlētāju noķer entītija, parādīsies izbīlis, un pēctam paziņos izdīvotais laiks, kā arī ziņojums, kā iet atpakaļ uz mapes izvēlni nospiežot R taustiņu.

Izvēloties Office, uzsākas spēle otrajā mapē.

32.att. Spēles uzsākšana otrajā mapē Office

Spēlētājam tiek paziņots, ka ir jaatrod gaismas lukturis.

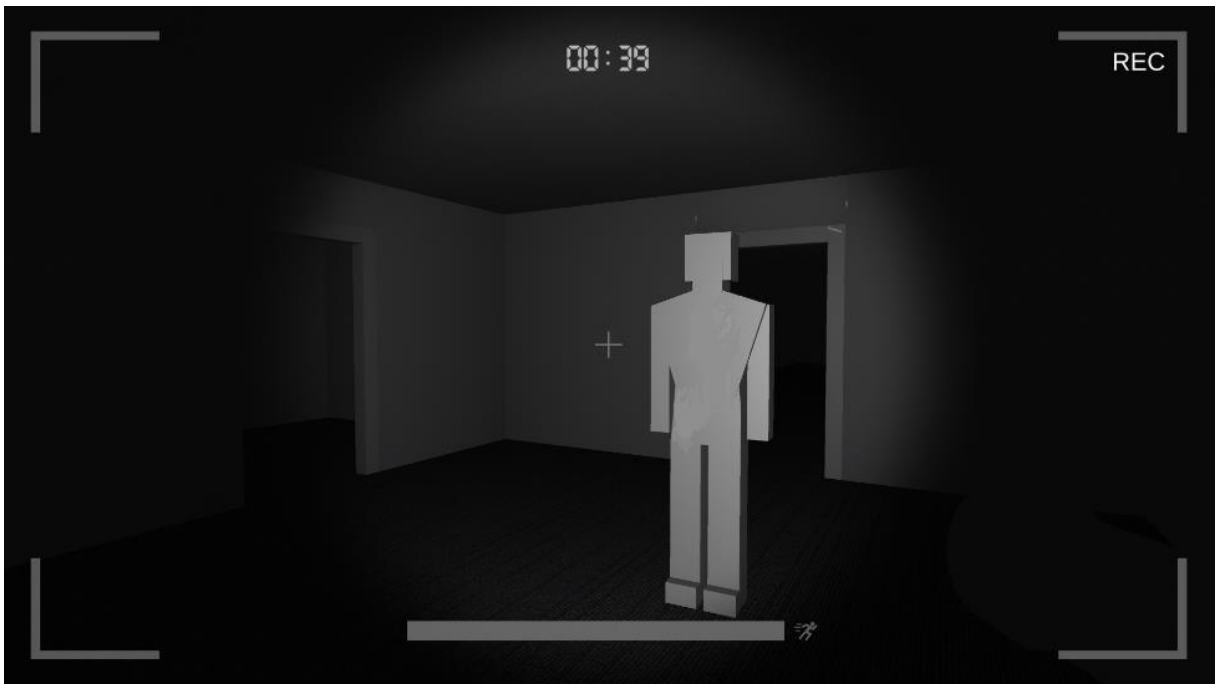


33.att. Iespējamā darbība ar gaismas lukturi

Paskatoties uz objektu, šajā situācijā - gaismas lukturi, tiks parādīts paziņojums, kā ar to darboties. (piem. “Nospied E, lai paņemtu”)

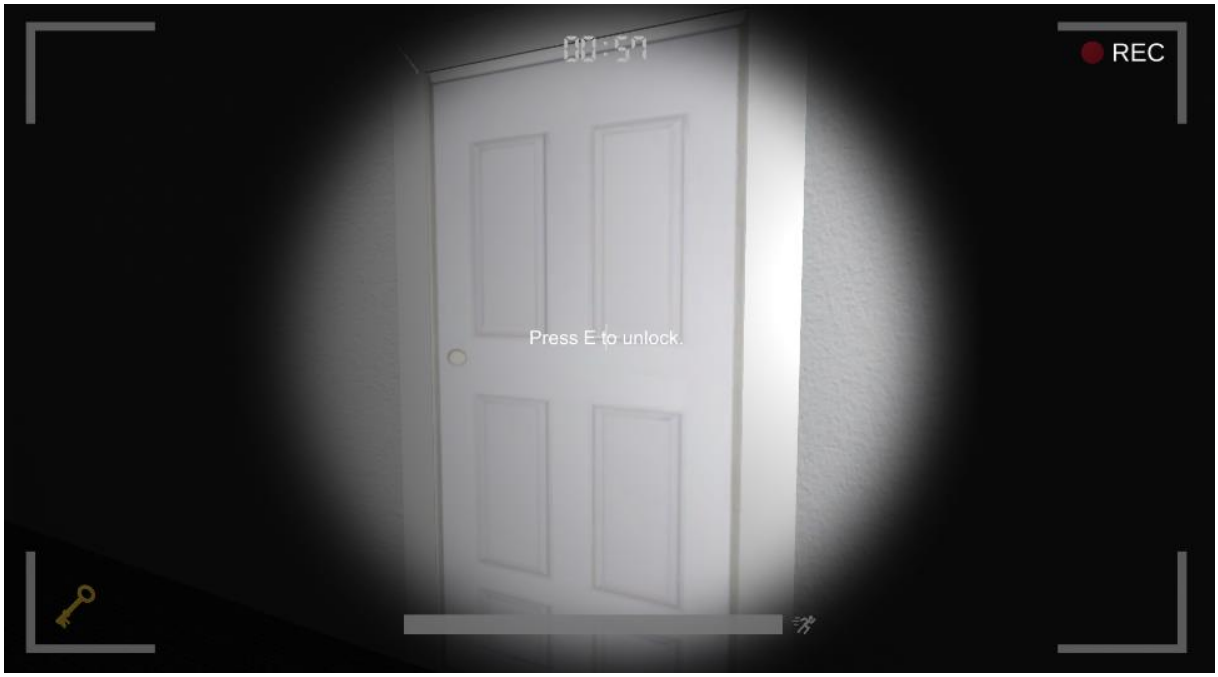
34.att. Iespējama darbība ar atslēgu

Paskatoties uz objektu, šajā situācija- atslēgu, tiks parādīts paziņojums, kā ar to darboties. (piem. “Nospied E, lai paņemtu”)



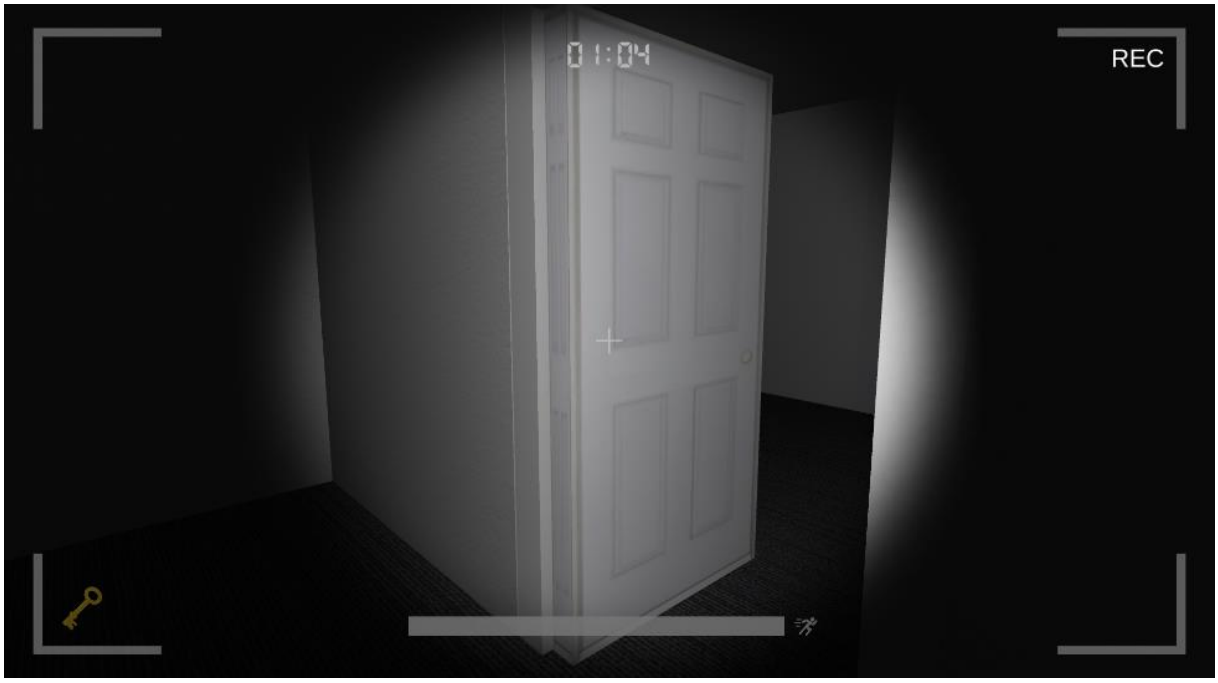
35.att. Entītija mapē otrajā mapē Office

Šī entītija sekos līzi spēlētājam, bet apstāsies ja uz to skatīsies. Spēlētājam ir jāizvairās no šīs entītijas, ja šī entītija noķers spēlētāju, spēle beigsies un tiks parādīts izbīlis zaudēšanas ekrāns. (skat. Att. 28. un 29.)



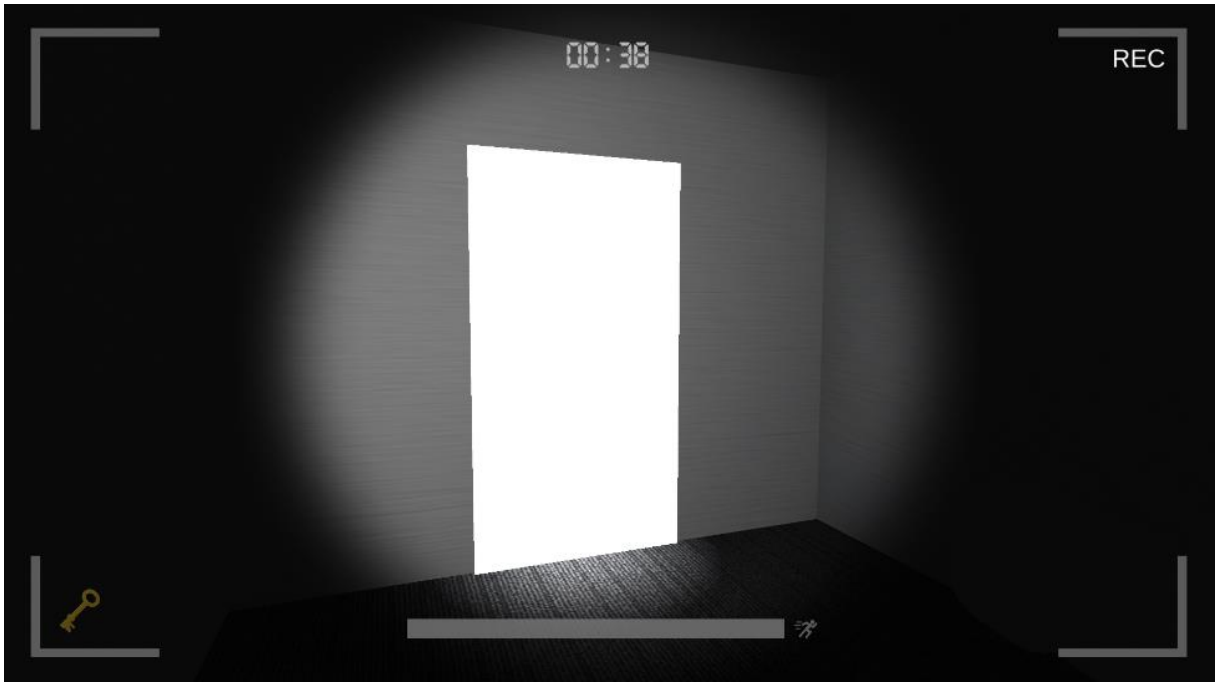
36.att. Iespējama darbība ar durvīm

Ja spēlētājam ir atslēga, tad ir iespējams atvērt durvis ar taustiņu E.



37.att. Atvērtas durvis

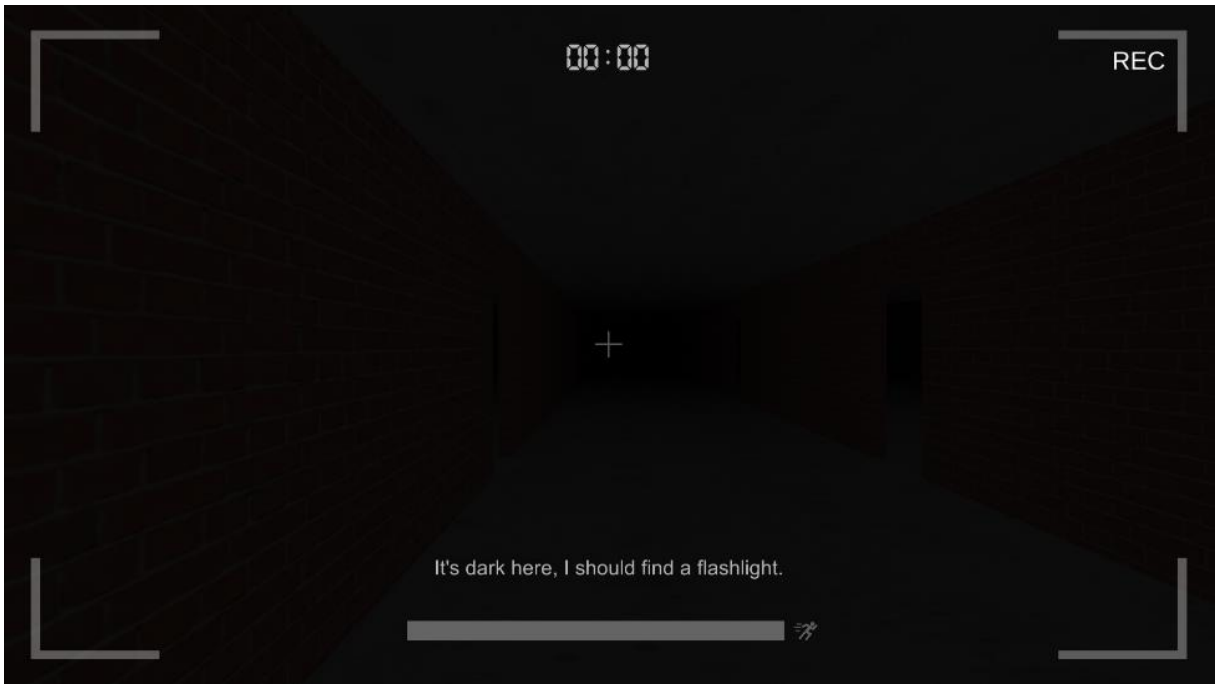
Pēc E taustiņa nospiešanas, durvis atverās.



38.att. Izeja pirmajā mapē Lobby

Spēlētājam saskaroties ar izeju, spēlētājs izbēg no mapes un uzvar spēli.

Izvēloties The Building, uzsākas spēle trešajā mapē.

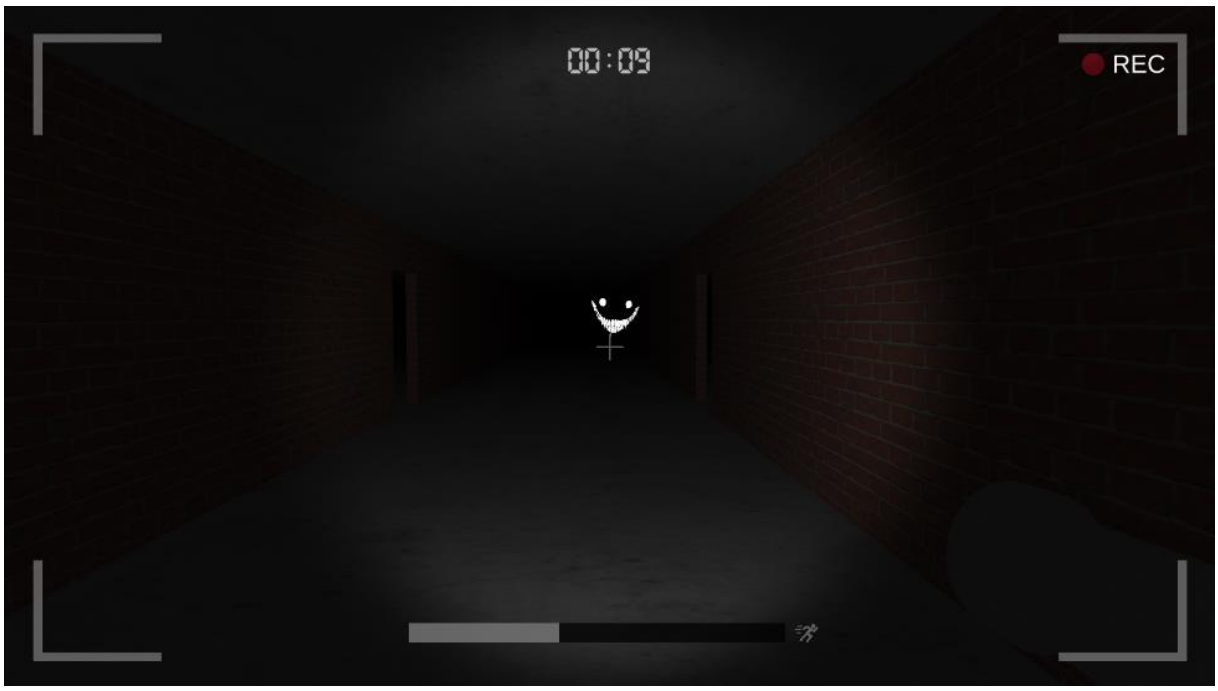


39.att. Spēles uzsākšana trešajā mapē The Building

Spēlētājam tiek paziņots, ka ir jaatrod gaismas lukturis.

40.att. Lukturis

Paskatoties uz objektu, šajā situācijā - gaismas lukturi, tiks parādīts paziņojums, kā ar to darboties. (piem. “Nospied E, lai paņemtu”)

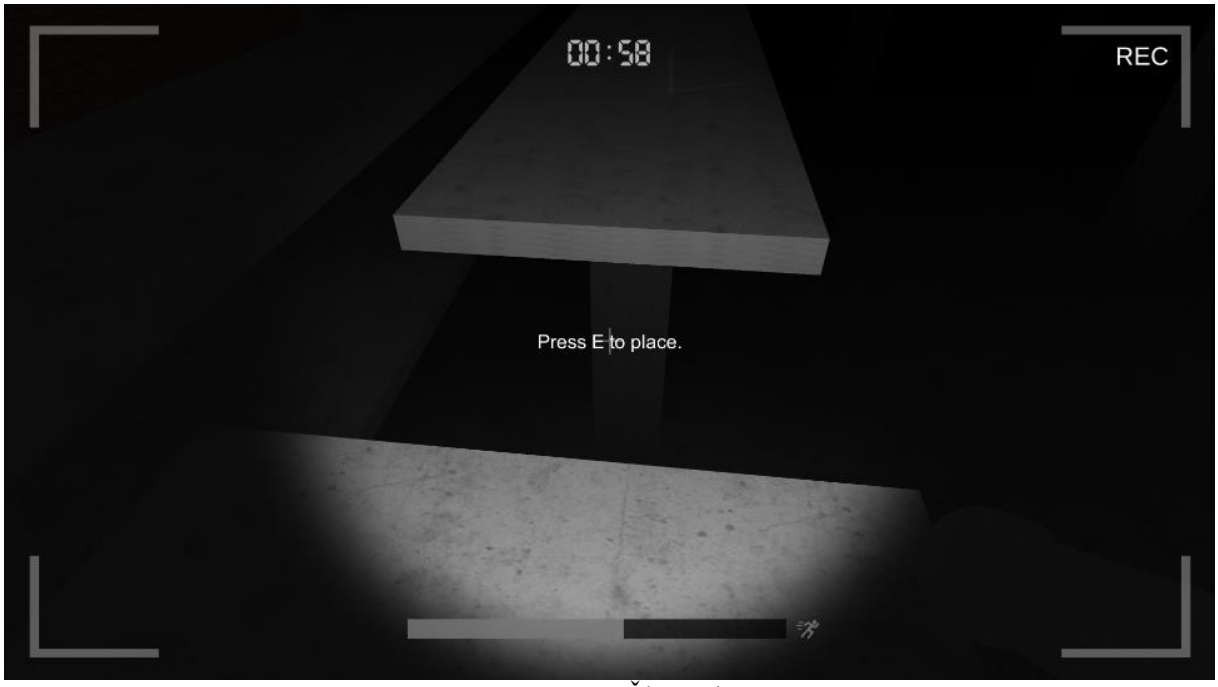


41.att. Entītija trešajā mapē The Building

Šīs entītijas parādīsies mapē un sekos līzi spēlētājam, bet pazudīs, ja uz to paskatīsies 2 sekundes. Spēlētājam ir jāizvairās no šīs entītijas, ja šī entītija noķers spēlētāju, spēle beigsies un tiks parādīts izbīlis zaudēšanas ekrāns. (skat. Att. 28. un 29.)

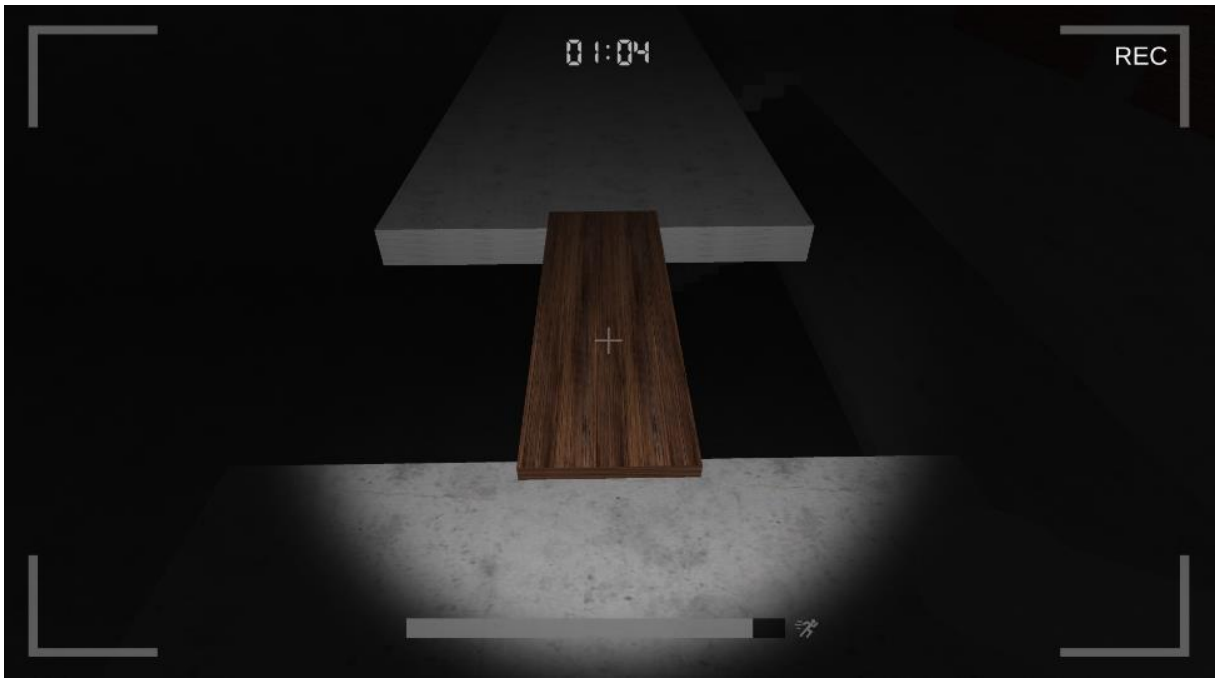
42.att. Nepaņemts dēlis

Paskatoties uz objektu, šajā situācijā - dēlis, tiks parādīts paziņojums, kā ar to darboties. (piem. “Nospied E, lai paņemtu”). Paņemot dēli, stūrī parādīsies tās ikona.



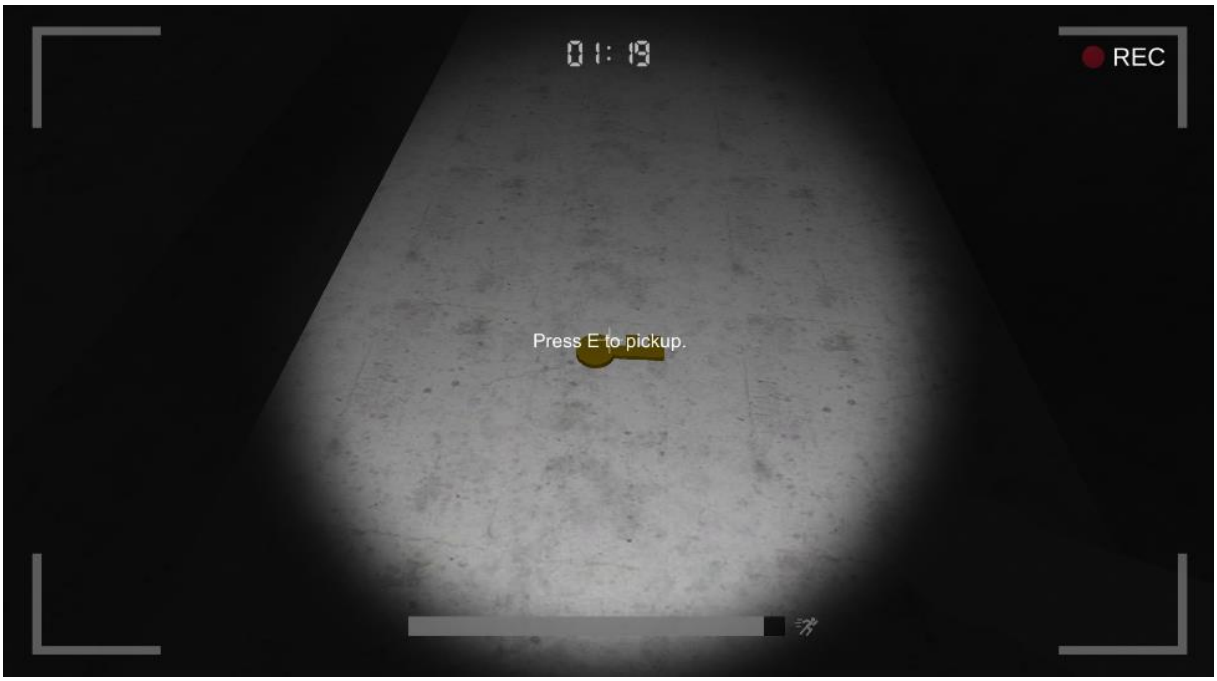
43.att. Šķērsojums

Ja spēlētājam ir dēlis, tad ir iespējams to nolikt ar taustiņu E. (piem. “Nospied E, lai noliktu (dēli).”)



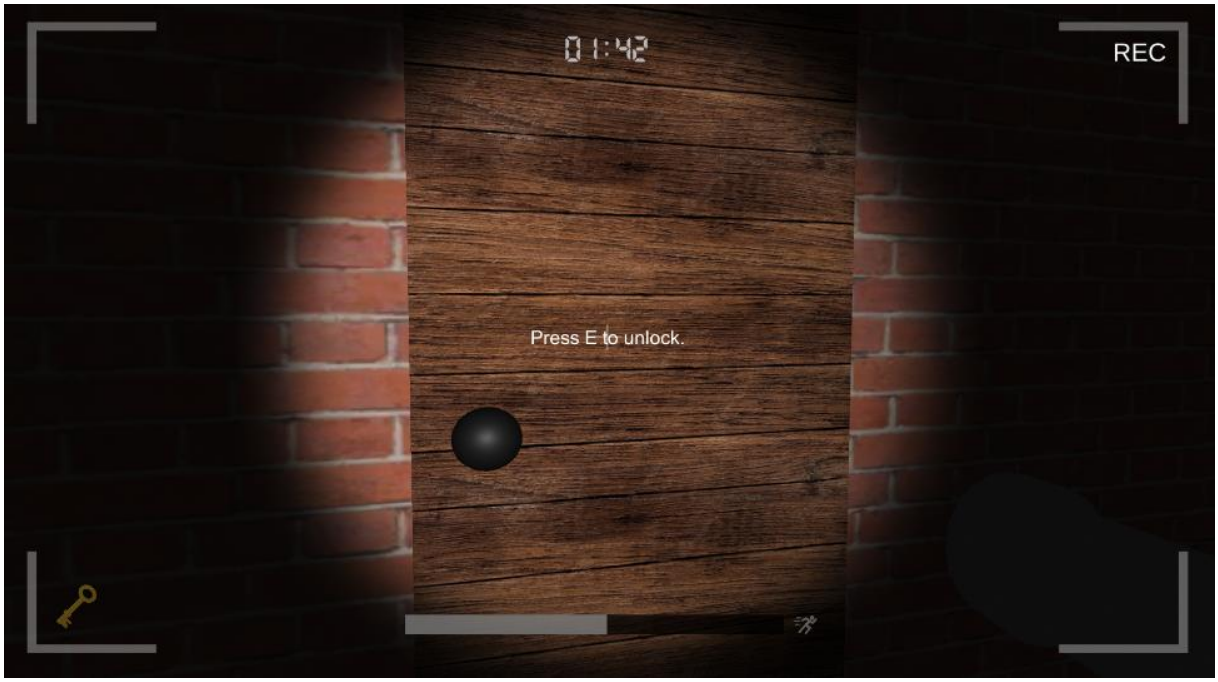
44.att. Nolikts dēlis

Pēc E taustiņa nospiešanas, dēlis tiek nolikts.



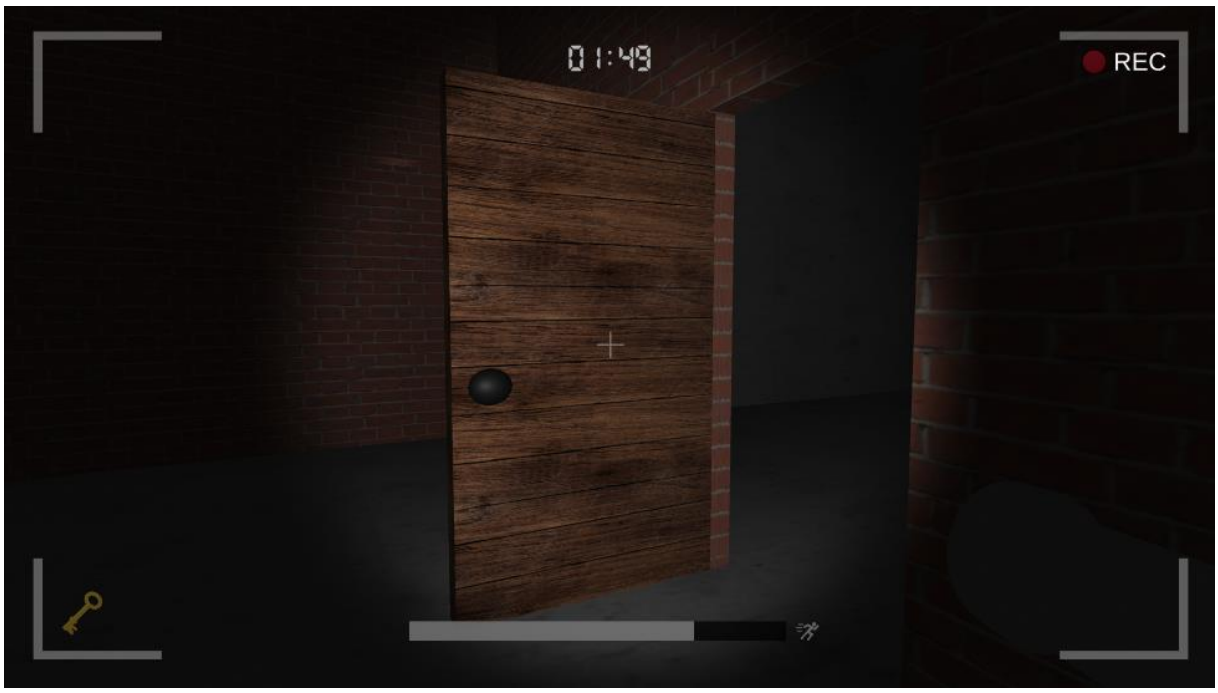
45.att. Paņemamas atslēgas

Paskatoties uz objektu, šajā situācija - atslēgu, tiks parādīts paziņojums, kā ar to darboties. (piem. “Nospied E, lai paņemtu”). Paņemot atslēgu, stūrī parādīsies tās ikona.



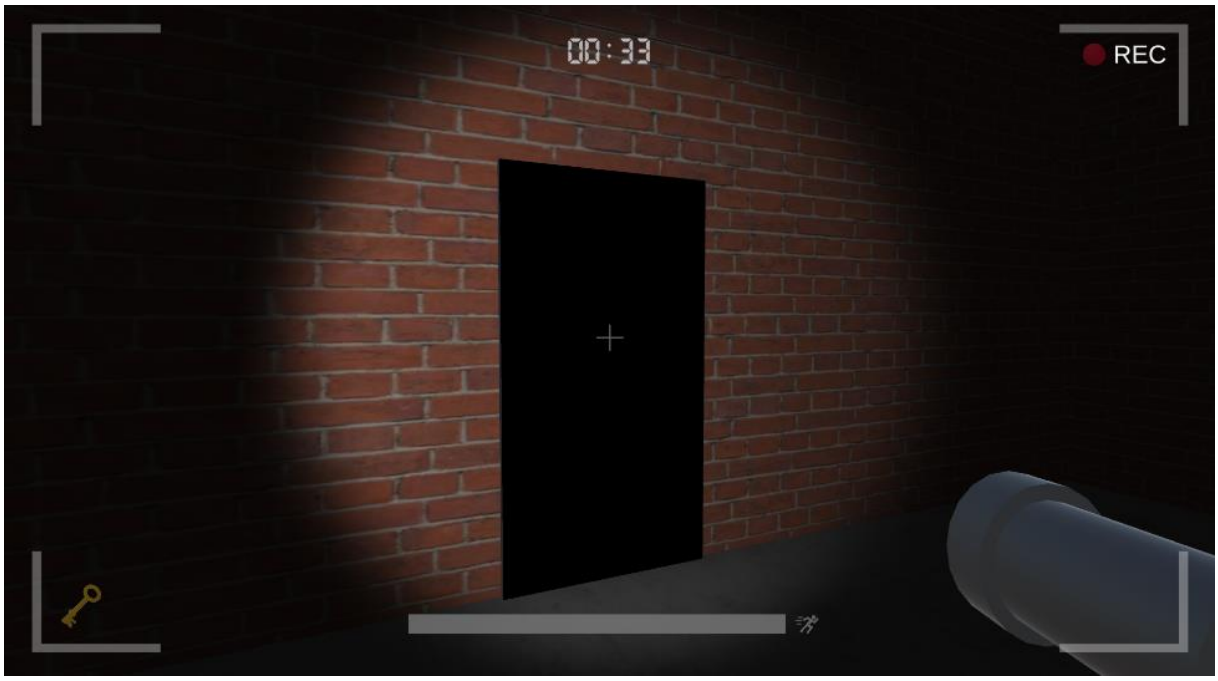
46.att. Iespējama darbība ar durvīm

Ja spēlētājam ir atslēga, tad ir iespējams atvērt durvis ar taustiņu E.



47.att. Atvērtas durvis

Pēc E taustiņa nospiešanas, durvis atverās.



48.att. Trešas mapes The Building izeja

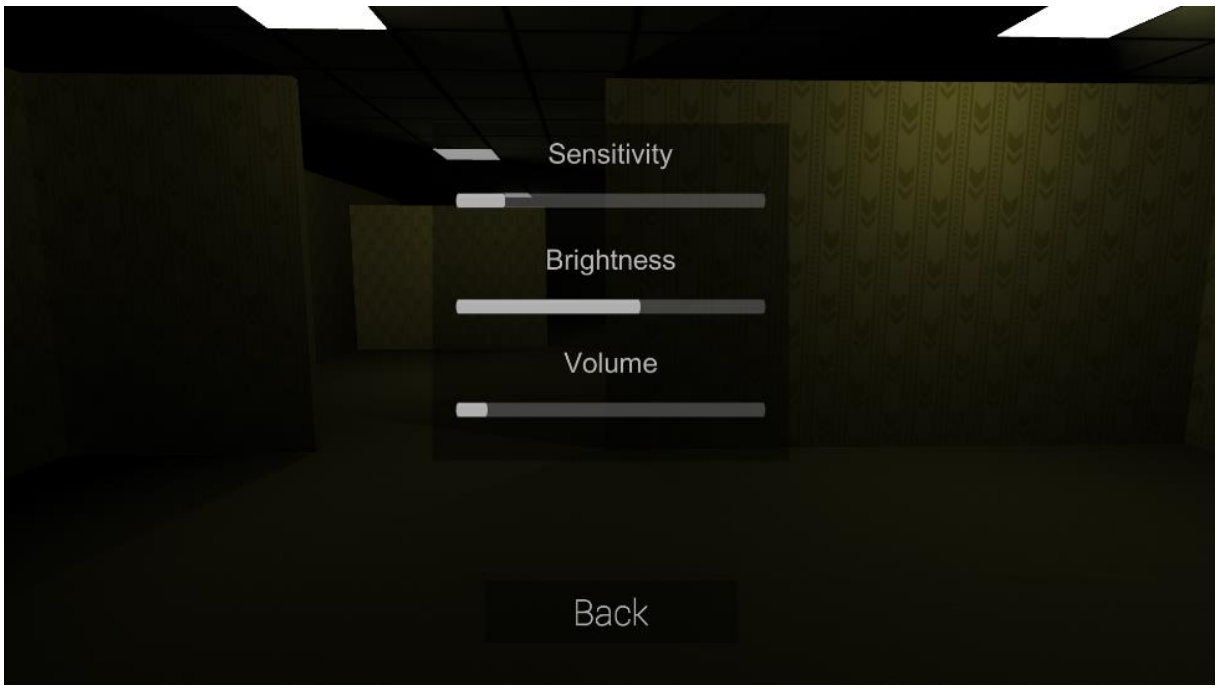
Spēlētājam saskaroties ar izeju, spēlētājs izbēg no mapes un uzvar spēli.

49.att. Opcijas izvēlne

Ja spēlētājs nospiež taustiņu Q, tiks atvērta opciju izvēlne.

Opciju izvēlnē ir 4 izvēles:

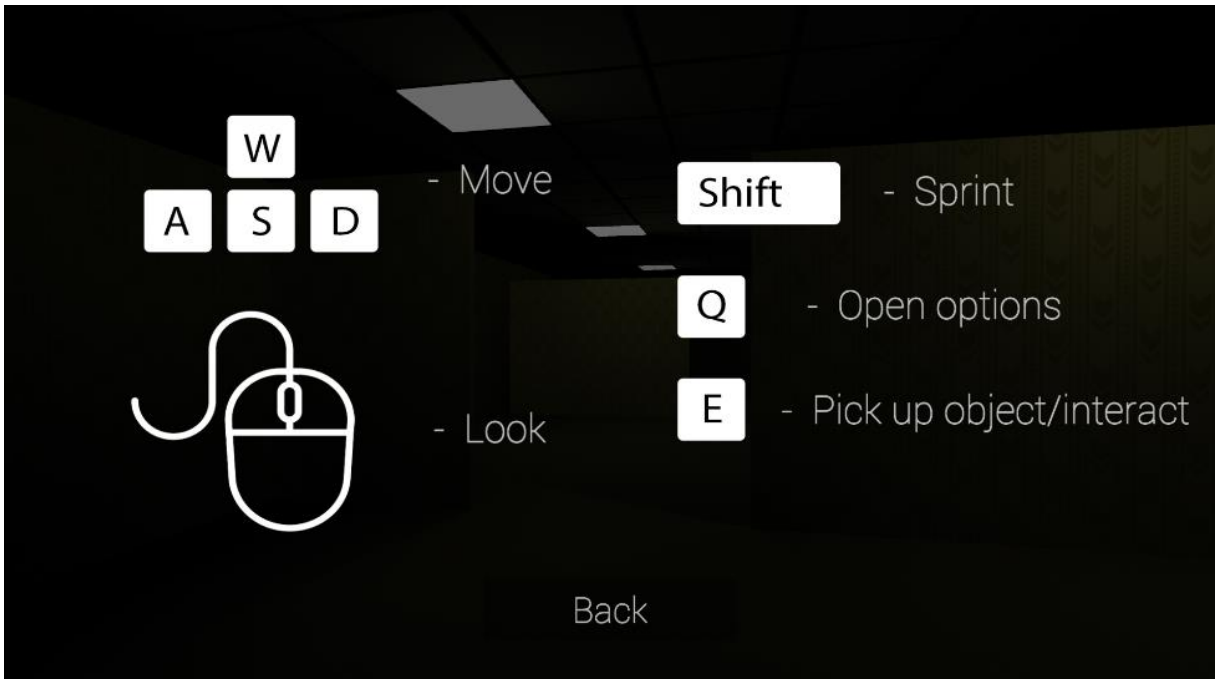
1. Resume, ko var noklikšķināt gan ar peli, gan ar taustiņu Q, kas ir atsākt spēli;
2. Settings, kas aizvedīs uz iestatījumu sadaļu;
3. Controls, kas aizvedīs uz kontroles pamācības sadaļu;
4. Main menu, kas aizvedīs uz Main Menu;



50.att. Iestatījumi

Iestatījumu sadaļā var redzēt 3 iestatījumus:

1. Sensitivity, kas regulē pelītes skatīšanās ātrumu (mouse sensivitity);
2. Brightness, kas regulē spēles gaišumu;
3. Volume, kas regulē spēles skaļumu;



51.att. Kontroles sadaļa

Šeit var apskatīt visas kontroles spēlei. Noklikšķinot Back, ies atpakal uz opciju izvēlni (skat. 47.att).

6.4. Testa piemērs

10. tabula

Nr.	Prasības numurs	Prasības nosaukums	Ievaddati/situācijas apraksts	Sagaidāmais rezultāts	Statuss
1.	2.1.	Jābut korektai opciju izvēlnes navigācijai.	Spēlētājs noklikšķina uz Main Menu	Tiek atvērta sadaļa Main Menu	Pareizi
2.			Spēlētājs noklikšķina uz Settings	Tiek atvērta sadaļa Settings	Pareizi
3.			Spēlētājs noklikšķina uz Resume	Tiek aizvērtas Opcijas	Pareizi
4.			Spēlētājs noklikšķina uz Back	Iziet no kādas opcijas sadaļas atpakal uz opcijas izvēlni	Pareizi
5.	2.10.	Konfigurējot Volume (spēles skaļums), mainās to valūta.	Spēlētājs kustina valūtas slīdni	Volume valūtas slīdnis vizuāli maina redzamo valūtu	Pareizi
6.			Jaunā valūta tiek saglabāta	Skriptā Volume valūta tiek nomainīta uz jauno	Pareizi
7.	4.1.	Apskatoties uz objektu, ar kuru var darboties, parādīt paziņojumu	Spēlētājs paskatās uz objektu.	Spēle reģistrē, ka spēlētājs skatās uz objektu	Pareizi
8.			Spēlētājs paskatās uz objektu, ar kuru var darboties.	Spēlētājam parāda ziņojumu, kā darboties ar objektu	Pareizi

NOBEIGUMS

Tika realizēta un izveidota 3D puzles šausmu spēle, ar intuitīvi interaktīvu lietotāja saskarni ar *Main Menu, Controls, How to play, Options* un *Settings* sadaļām, iestatījumiem, kurus var pielāgot jebkurš lietotājs un uzlabot savu spēlēs pieredzi, 3 dažādām paštaisītām mapēm, kur katra atšķiras dizainā un grūtības pakāpe, un mapei unikālām baismīgām entītijām, no kurām spēlētājam ir jāizdzīvo, mēģinot izprast, kā no tām izbēgt, kā arī priekšmetiem, un mapes objektiem, kurus spēlētājam ir jāsavāc un jāatrod, lai varētu izbēgt no mapes un uzvarēt spēli, bet ja sliktākajā gadījumā spēlētāju noķer kāda no entītijām, tad nobiedēt spēlētāju ar izbīli un skaņu, beidzot spēli.

Projekts tika veidots *Unity* spēļu izveides aplikācijā, programmējot C# programmēšanas valodā. Projekts tikai pabeigts veiksmīgi ar veiksmīgiem rezultātiem un pareizi strādājošām funkcijām.

Tā kā veidotais projekts ir datorspēle, tai ir neskaitāmi daudz iespējamu jaunu funkciju un jaunumu ieviešana.

Tuvākajā nākotnē spēli varēs izmēģināt jebkurš. Tā bus atrodama publiskajā spēļu dalīšanās tīmekļa vietnē https://itch.io/.

INFORMĀCIJAS AVOTI

1. Kvalifikācijas eksāmena programma - https://www.visc.gov.lv/lv/media/23847/download?attachment
2. RVT e-studijas pamācošās prezentācijas - https://e.rvt.lv/course/view.php?id=633#section-4
3. RVT e-studijas pamācošais kurss "Programmatūras dokumentēšana dažādās tās attīstības stadijās" - https://e.rvt.lv/course/view.php?id=633#section-2
4. Apguvu, *Unity* pamatus - https://learn.unity.com/
5. Apguvu, kā lietot *Unity* - https://docs.unity3d.com/Manual/
6. Apguvu, kā lietot *Unity* saskarni - https://docs.unity3d.com/Manual/UnityOverview.html
7. Apguvu, kā darboties ar *Unity UI* - https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html
8. Apguvu, kā darboties ar *Unity* skriptošanu - https://docs.unity3d.com/Manual/ScriptingSection.html
9. Risināju problēmas saistītas ar projekta veidošanu - https://discussions.unity.com/
10. Risināju problēmas saistītas ar projekta veidošanu - https://stackoverflow.com/
11. Risināju problēmas saistītas ar projekta veidošanu - https://gamedev.stackexchange.com/
12. Tika meklēti audio - https://pixabay.com/
13. Tika meklēti audio - https://www.epidemicsound.com/
14. Tika meklēti audio - https://mixkit.co/free-sound-effects/horror/
15. Tika ņemts kods prieks *StaminaController.cs* - https://www.youtube.com/watch?v=Fs2YCoamO_U&ab_channel=SpeedTutor
16. Tika ņemti un rediģēti kodi FirstPersonControllerCustom.cs, KeyScript.cs, Door.cs - https://www.youtube.com/watch?v=g2cEOHzMDBc&list=PLWI8H56cvVoI0xd2FOIDschZbTKXlm71r&ab_channel=Omogonix

PIELIKUMI

1. pielikums

BrightnessScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Rendering.PostProcessing;

public class BrightnessScript : MonoBehaviour
{
    public Slider BrightnessSlider;

    public PostProcessProfile Brightness;
    public PostProcessLayer Layer;

    AutoExposure exposure;

    private const string BrightnessKey = "Brightness";

    void Start()
    {

        Brightness.TryGetSettings(out exposure);

        float savedBrightness = PlayerPrefs.GetFloat(BrightnessKey, 0.5f);
        BrightnessSlider.value = savedBrightness;

        AdjustBrightness(savedBrightness);
    }

    public void AdjustBrightness(float value)
    {
        if (value != 0)
        {
            exposure.keyValue.value = value;

            PlayerPrefs.SetFloat(BrightnessKey, value);
        }
        else
```

```
            {
                exposure.keyValue.value = .05f;

                PlayerPrefs.SetFloat(BrightnessKey, .05f);
            }
        }
    }
```

ButtonHover.cs

```
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

public class ButtonHover : MonoBehaviour, IPointerEnterHandler,
IEventSystemHandler, IPointerExitHandler
{
public Text PlayText;

public AudioSource hover;

public void OnPointerEnter(PointerEventData eventData)
{
	PlayText.color = Color.white;
	hover.Play();
}

public void OnPointerExit(PointerEventData eventData)
{
	PlayText.color = Color.grey;
}
}
```

ButtonOption.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityStandardAssets.Characters.FirstPerson;

public class ButtonOption : MonoBehaviour
{

public FirstPersonControllerCustom playerController;
public PauseScript PauseScript;

private void Start()
{
	Cursor.visible = true;
	Cursor.lockState = CursorLockMode.None;
}

public void MapSelect()
{
	SceneManager.LoadScene(1);
}

public void MainMenu()
{
	SceneManager.LoadScene(0);
}

public void Map1()
{
	SceneManager.LoadScene(2);
	Time.timeScale = 1;
	playerController.canMove = true;
	PauseScript.Resume();

}

public void Map2()
{
	SceneManager.LoadScene(3);
```

```
        Time.timeScale = 1;
        playerController.canMove = true;
        PauseScript.Resume();
}

public void Map3()
{
        SceneManager.LoadScene(4);
        Time.timeScale = 1;
        playerController.canMove = true;
        PauseScript.Resume();
}

public void Controls()
{
        SceneManager.LoadScene(7);
        PauseScript.Resume();
}

public void Tutorial()
{
        SceneManager.LoadScene(8);
        PauseScript.Resume();
}

public void Exit()
{
        Application.Quit();
}
}
```

ControlsPanelScript.cs

```
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;


public class ControlsPanelScript : MonoBehaviour
{
    public GameObject PauseMenu, SettingsMenu, ControlsMenu;
    public FirstPersonControllerCustom playerController;
    public GameObject CameraOverlay, StaminaCanvas;


    void Start()
    {

    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Q))
        {
            if (ControlsMenu.activeSelf)
            {
                Resume();

            }
        }


    }


    public void Resume()
    {
        PauseMenu.SetActive(false);
        SettingsMenu.SetActive(false);
        ControlsMenu.SetActive(false);
        Time.timeScale = 1;
        Cursor.lockState = CursorLockMode.Locked;
```

```
            Cursor.visible = false;
            playerController.canMove = true;
            CameraOverlay.SetActive(true);
            StaminaCanvas.SetActive(true);
        }

    }
```

defeatscreen.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using TMPro;

public class defeatscreen : MonoBehaviour
{

    [SerializeField] TextMeshProUGUI elapsedTimeText;


    void Start()
    {
        float elapsedTime = PlayerPrefs.GetFloat("ElapsedTime", 0f);
        int minutes = Mathf.FloorToInt(elapsedTime / 60);
        int seconds = Mathf.FloorToInt(elapsedTime % 60);

        elapsedTimeText.text = string.Format("You survived for {0:00}:{1:00}",
minutes, seconds);

    }

    private void Update()
{
       if (Input.GetKey(KeyCode.R))
       {
              SceneManager.LoadScene(0);
       }
       if (Input.GetKey(KeyCode.Escape))
       {
              Application.Quit();
       }

}
}
```

Door.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class jumpscare : MonoBehaviour
{
public string scenename;

private void OnTriggerEnter(Collider other)
{
	if (other.CompareTag("MainCamera"))
	{
		SceneManager.LoadScene(scenename);
	}
}
}
```

DotScript.cs

```
using UnityEngine;
using UnityEngine.UI;

public class DotScript : MonoBehaviour
{
    public float beepInterval = 0.5f;
    private Image DotImage;

    void Start()
    {
        DotImage = GetComponent<Image>();
        InvokeRepeating("ToggleVisibility", 0f, beepInterval);
    }

    void ToggleVisibility()
    {
        DotImage.enabled = !DotImage.enabled;
    }
}
```

ElapsedTime.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using TMPro;

public class ElapsedTime : MonoBehaviour
{

    [SerializeField] TextMeshProUGUI elapsedTimeText;

    void Start()
    {
        float elapsedTime = PlayerPrefs.GetFloat("ElapsedTime", 0f);

        int minutes = Mathf.FloorToInt(elapsedTime / 60);
        int seconds = Mathf.FloorToInt(elapsedTime % 60);

        string minutesText = (minutes == 1) ? "minute" : "minutes";

        string secondsText = (seconds == 1) ? "second" : "seconds";

        if (minutes > 0)
        {
            elapsedTimeText.text = string.Format("You escaped in {0} {1} and {2}
{3}", minutes, minutesText, seconds, secondsText);
        }
        else
        {
            elapsedTimeText.text  =  string.Format("You  escaped  in  {0}  {1}",
seconds, secondsText);
        }
    }
}
```

exitgame.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class exitgame : MonoBehaviour
{

private void Update()
{
	if (Input.GetKey(KeyCode.Escape))
	{
		SceneManager.LoadScene(0);
	}
}
}
```

ExitScript.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ExitScript : MonoBehaviour
{
public string scenename;

public GameObject exittext;

public GameObject player;

private void OnTriggerStay(Collider other)
{
	if (other.CompareTag("MainCamera"))
	{
		exittext.SetActive(value: true);
		if (Input.GetKey(KeyCode.E))
		{
			SceneManager.LoadScene(scenename);
		}
	}
}

private void OnTriggerExit(Collider other)
{
	if (other.CompareTag("MainCamera"))
	{
		exittext.SetActive(value: false);
	}
}
}
```

FirstPersonControllerCustom.cs

```
using UnityEngine;
using UnityEngine.UI;


namespace UnityStandardAssets.Characters.FirstPerson
{
    public class FirstPersonControllerCustom : MonoBehaviour
    {
        public float walkingSpeed = 9f;
        public float runningSpeed = 112.5f;
        public float jumpSpeed = 8f;
        public float gravity = 20f;
        public Camera playerCamera;
        public float lookSpeed = 100f;
        public float lookXLimit = 45f;


        private CharacterController characterController;
        private Vector3 moveDirection = Vector3.zero;
        private float rotationX;

        [HideInInspector]
        public bool canMove = true;

        [HideInInspector]
        public StaminaController _staminaController;

        public AudioClip walkingSound;
        private AudioSource audioSource;

        public bool CanMove
        {
            get { return canMove; }
            set { canMove = value; }
        }

        public Slider SensitivitySlider;
```

```
private const string SensitivityKey = "Sensitivity";

private void Start()
{
    _staminaController = GetComponent<StaminaController>();
    characterController = GetComponent<CharacterController>();
    Cursor.lockState = CursorLockMode.Locked;
    Cursor.visible = false;



    lookSpeed = PlayerPrefs.GetFloat(SensitivityKey, 4f);


    SensitivitySlider.value = lookSpeed;
}

public void AdjustSpeed(float newSpeed)
{
    lookSpeed = newSpeed * 1;


    PlayerPrefs.SetFloat(SensitivityKey, newSpeed);
}

public void SetRunSpeed(float speed)
{
    runningSpeed = speed;
}



private void Update()
{
    Vector3 vector = transform.TransformDirection(Vector3.forward);
    Vector3 vector2 = transform.TransformDirection(Vector3.right);
    bool key = Input.GetKey(KeyCode.LeftShift);

    if (!key)
    {
        _staminaController.weAreSprinting = false;
    }
```

```
                if (key && _staminaController.playerStamina > 0f)
                {
                    _staminaController.weAreSprinting = true;
                    _staminaController.Sprinting();
                }

                float num = (canMove ? ((key ? runningSpeed : walkingSpeed) *
Input.GetAxis("Vertical")) : 0f);
                float num2 = (canMove ? ((key ? runningSpeed : walkingSpeed) *
Input.GetAxis("Horizontal")) : 0f);
                float y = moveDirection.y;

                moveDirection = vector * num + vector2 * num2;




                if (Input.GetButton("Jump") && canMove &&
characterController.isGrounded)
                {
                    moveDirection.y = jumpSpeed;
                }
                else
                {
                    moveDirection.y = y;
                }

                if (!characterController.isGrounded)
                {
                    moveDirection.y -= gravity * Time.deltaTime;
                }

                characterController.Move(moveDirection * Time.deltaTime);

                if (canMove)
                {
                    rotationX += (-Input.GetAxis("Mouse Y")) * lookSpeed;
                    rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit);
                    playerCamera.transform.localRotation =
Quaternion.Euler(rotationX, 0f, 0f);
                    transform.rotation *= Quaternion.Euler(0f, Input.GetAxis("Mouse
X") * lookSpeed, 0f);
                }
```

```
            }
        }
    }
```

FlashlightPickup.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FlashlightPickup : MonoBehaviour
{
    public GameObject inticon, flashlight_ground, flashlight_hand,
Flashlight_held, FlashlightInHand;
    private bool PickedUp = false;
    public AudioSource pickup;



    void Update()
    {
        if (inticon.activeSelf && Input.GetKeyDown(KeyCode.E) && !PickedUp)
        {
            PickUpFlashlight();
        }


    }


    void OnTriggerStay(Collider other)
    {
        if (other.CompareTag("MainCamera"))
        {
            inticon.SetActive(true);
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("MainCamera"))
        {
            inticon.SetActive(false);
        }
```

```
    }

    void PickUpFlashlight()
    {
        flashlight_ground.SetActive(false);
        inticon.SetActive(false);
        flashlight_hand.SetActive(true);
        Flashlight_held.SetActive(true);
        PickedUp = true;
        pickup.Play();
        Debug.Log("Flashlight picked up");

    }


}
```

FreezeMonsterScript.cs

```
using UnityEngine;

public class FreezeMonsterScript : MonoBehaviour
{
    public Rigidbody monsRigid;
    public Transform monsTrans;
    public Transform playTrans;
    public int monsSpeed;
    public float stopDistance = 2f;

    private void FixedUpdate()
    {
        Vector3 directionToPlayer = (playTrans.position - monsTrans.position).normalized;

        if (IsPlayerLookingAtMonster())
        {
            monsRigid.velocity = Vector3.zero;
        }
        else
        {
            monsRigid.velocity = directionToPlayer * monsSpeed * Time.deltaTime;
        }

        monsTrans.LookAt(playTrans);
    }

    private bool IsPlayerLookingAtMonster()
    {
        //calculates players direction from the monster
        Vector3 directionToMonster = (monsTrans.position - playTrans.position).normalized;

        //calculates the angle of players looking dot relative to the monster
        float dotProduct = Vector3.Dot(playTrans.forward, directionToMonster);

        //check the angle to determine if player is looking at the monster
        return dotProduct > 0.5f;
```

```
        }
    }
```

FreezeNavMeshScript.cs

```
using UnityEngine;
using UnityEngine.AI;

public class FreezeNavMeshScript : MonoBehaviour
{
    [SerializeField] private Transform playerTransform;
    [SerializeField] private float stopDistance = 2f;

    private NavMeshAgent navMeshAgent;

    private void Awake()
    {
        navMeshAgent = GetComponent<NavMeshAgent>();
    }

    private void Update()
    {
        if (playerTransform != null)
        {
            navMeshAgent.destination = playerTransform.position;

            float distanceToPlayer = Vector3.Distance(transform.position,
playerTransform.position);
            if (distanceToPlayer < stopDistance && IsPlayerLookingAtObject())
            {
                navMeshAgent.isStopped = true;
            }
            else
            {
                navMeshAgent.isStopped = false;
            }
        }
    }

    private bool IsPlayerLookingAtObject()
    {
        Vector3 directionToObject = transform.position -
Camera.main.transform.position;
```

```
            return Vector3.Dot(Camera.main.transform.forward, directionToObject) > 0;
        }
    }
```

jumpscare.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class jumpscare : MonoBehaviour
{
public string scenename;

private void OnTriggerEnter(Collider other)
{
	if (other.CompareTag("MainCamera"))
	{
		SceneManager.LoadScene(scenename);
	}
}
}
```

KeyScript.cs

```
using UnityEngine;
using UnityEngine.UI;

public class KeyScript : MonoBehaviour
{
    public GameObject inticon_k, key, Icon;
    public AudioSource pickup, warning;
    public GameObject LurkingMonster;
    public GameObject PresenceText;
    public float fadeDuration;

    private void Start()
    {

        SetTextAlpha(0f);
    }

    private void OnTriggerStay(Collider other)
    {
        if (other.CompareTag("MainCamera"))
        {
            inticon_k.SetActive(true);

            if (Input.GetKey(KeyCode.E))
            {
                key.SetActive(false);
                Door.keyfound = true;
                inticon_k.SetActive(false);
                Icon.SetActive(true);
                pickup.Play();
                warning.Play();
                LurkingMonster.SetActive(true);


                ShowText("You feel a presence following you...", fadeDuration);
            }
        }
    }
```

```
    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("MainCamera"))
        {
            inticon_k.SetActive(false);
        }
    }


    private void ShowText(string text, float duration)
    {
        PresenceText.SetActive(true);
        PresenceText.GetComponent<Text>().text = text;
        PresenceText.GetComponent<Text>().CrossFadeAlpha(1f,       fadeDuration,
false);
        Invoke("HideText", duration);
    }


    private void HideText()
    {
        PresenceText.GetComponent<Text>().CrossFadeAlpha(0f,       fadeDuration,
false);
        Invoke("DeactivateText", fadeDuration);
    }


    private void DeactivateText()
    {
        PresenceText.SetActive(false);
    }


    private void SetTextAlpha(float alpha)
    {
        PresenceText.GetComponent<Text>().canvasRenderer.SetAlpha(alpha);
    }

}
```

MenuButtons.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityStandardAssets.Characters.FirstPerson;


public class MenuButtons : MonoBehaviour
{
public GameObject PauseMenu, SettingsMenu, ControlsMenu;
public FirstPersonControllerCustom playerController;
public GameObject CameraOverlay, StaminaCanvas;


public void Resume()
{
        Debug.Log("Resumed");
         PauseMenu.SetActive(false);
        SettingsMenu.SetActive(false);
        Time.timeScale = 1;
         Cursor.lockState = CursorLockMode.Locked;
         Cursor.visible = false;
        playerController.canMove = true;
        CameraOverlay.SetActive(true);
        StaminaCanvas.SetActive(true);
}

public void MapSelect()
{
        SceneManager.LoadScene(1);
}


public void MainMenu()
{
        SceneManager.LoadScene(0);
}

public void Settings()
    {
        PauseMenu.SetActive(false);
```

```
        SettingsMenu.SetActive(true);
    }

    public void Back()
    {
        PauseMenu.SetActive(true);
        SettingsMenu.SetActive(false);
        ControlsMenu.SetActive(false);
    }

    public void Controls()
    {
        ControlsMenu.SetActive(true);
        PauseMenu.SetActive(false);
        SettingsMenu.SetActive(false);
    }
    }
```

menuscript.cs

```
using UnityEngine;

public class menuscript : MonoBehaviour
{
private void Update()
{
	if (Input.GetKey(KeyCode.Escape))
	{
		Application.Quit();
	}
}
}
```

Monologue.cs

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class Monologue : MonoBehaviour
{
    public Text textElement;
    public float displayDuration = 7.0f;

    void Start()
    {
        StartCoroutine(DisplayText());
    }

    IEnumerator DisplayText()
    {
        textElement.enabled = true;

        yield return new WaitForSeconds(displayDuration);

        textElement.enabled = false;
    }
}
```

MonsterScript.cs

```
using UnityEngine;

public class MonsterScript : MonoBehaviour
{
public Rigidbody monsRigid;

public Transform monsTrans;

public Transform playTrans;

public int monsSpeed;

private void FixedUpdate()
{
	monsRigid.velocity = base.transform.forward * monsSpeed * Time.deltaTime;
}

private void Update()
{
	monsTrans.LookAt(playTrans);
}
}
```

MovementScripts.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MovementScripts : MonoBehaviour
{
    public GameObject PauseMenu;

    public GameObject footstep;
    public GameObject runningFootstep;
    private bool FootstepActive = false;
    private bool RunningFootstepActive = false;

    public Transform cameraTransform;

    private Vector3 initialCameraPosition;

    private StaminaController staminaController;

    public float walkingBobbingSpeed = 10f;
    public float walkingBobbingAmount = 0.05f;
    public float runningBobbingSpeed = 15f;
    public float runningBobbingAmount = 0.1f;

    void Start()
    {

        footstep.SetActive(false);
        runningFootstep.SetActive(false);
        staminaController = GetComponent<StaminaController>();
        cameraTransform = GetComponentInChildren<Camera>().transform;
        initialCameraPosition = cameraTransform.localPosition;
    }

    void Update()
    {
```

```
            bool walkingKeyPressed = Input.GetKey("w") || Input.GetKey("s") ||
Input.GetKey("a") || Input.GetKey("d"); //Determines if player is moving or sprinting
            bool shiftKeyPressed = Input.GetKey(KeyCode.LeftShift);

            if (walkingKeyPressed) //If walking, then make footstep sounds and do
walking camera bobbing
            {
                footsteps();
                WalkingCameraBobbing();
            }
            else if (FootstepActive)
            {
                StopFootsteps();
                ResetCameraPosition();
            }

            if (staminaController.playerStamina > 0f)
            {
                if (walkingKeyPressed && shiftKeyPressed) //If running, then make
running sounds and do running camera bobbing
                {
                    StopFootsteps();
                    runningFootsteps();
                    RunningCameraBobbing();
                }
                else if (RunningFootstepActive)
                {
                    StopRunningFootsteps();
                    ResetCameraPosition();
                }
            }
            else
            {
                StopRunningFootsteps();
                ResetCameraPosition();
            }

            if (staminaController.playerStamina == 0f)
            {
                StopRunningFootsteps();
                ResetCameraPosition();
            }
        }
```

```
            void footsteps()
            {
                footstep.SetActive(true);
                FootstepActive = true;
            }

            void StopFootsteps()
            {
                footstep.SetActive(false);
                FootstepActive = false;
            }

            void runningFootsteps()
            {
                runningFootstep.SetActive(true);
                RunningFootstepActive = true;
            }

            void StopRunningFootsteps()
            {
                runningFootstep.SetActive(false);
                RunningFootstepActive = false;
            }

            void WalkingCameraBobbing() //Formula for walking camera movement
            {
                float bobbingOffset = Mathf.Sin(Time.time * walkingBobbingSpeed) *
walkingBobbingAmount;
                Vector3 bobbingPosition = initialCameraPosition;
                bobbingPosition.y += bobbingOffset;
                cameraTransform.localPosition = bobbingPosition;
            }

            void RunningCameraBobbing() //Formula for running camera movement
            {
                float bobbingOffset = Mathf.Sin(Time.time * runningBobbingSpeed) *
runningBobbingAmount;
                Vector3 bobbingPosition = initialCameraPosition;
                bobbingPosition.y += bobbingOffset;
                cameraTransform.localPosition = bobbingPosition;
            }
```

```
            void ResetCameraPosition()
            {
                cameraTransform.localPosition = initialCameraPosition; //Puts camera back
in original place
            }
        }
```

NavMeshScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class NavMeshScript : MonoBehaviour
{
    [SerializeField] private Transform movePositionTransform;

    private NavMeshAgent navMeshAgent;

    private void Awake()
    {
        navMeshAgent = GetComponent<NavMeshAgent>();
    }

    private void Update()
    {
        navMeshAgent.destination = movePositionTransform.position;
    }
}
```

PauseScript.cs

```
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;


public class PauseScript : MonoBehaviour
{
    public GameObject PauseMenu;
    public FirstPersonControllerCustom playerController;
    public GameObject CameraOverlay, StaminaCanvas;

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Q))
        {
            if (!PauseMenu.activeSelf)
            {
                Pause();
            }
            else
            {
                Resume();
            }
        }
    }

    public void Pause()
    {
        PauseMenu.SetActive(true);
        Time.timeScale = 0;
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
        playerController.canMove = false;

        CameraOverlay.SetActive(false);
        StaminaCanvas.SetActive(false);
    }

    public void Resume()
```

```
    {
        PauseMenu.SetActive(false);
        Time.timeScale = 1;
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
        playerController.canMove = true;

        CameraOverlay.SetActive(true);
        StaminaCanvas.SetActive(true);
    }
}
```

Plank.cs

```
using UnityEngine;

public class Plank : MonoBehaviour
{
public GameObject plank_unplaced;

public GameObject plank_placed;

public GameObject intText;

public GameObject lockedtext;

public AudioSource open;

public AudioSource close;

public bool opened;

public bool locked;

public static bool keyfound;

private void Start()
{
	keyfound = false;
}

private void OnTriggerStay(Collider other)
{
	if (!other.CompareTag("MainCamera") || opened)
	{
		return;
	}
	if (!locked)
	{
		intText.SetActive(value: true);
		if (Input.GetKey(KeyCode.E))
		{
```

```
                    plank_unplaced.SetActive(value: false);
                    plank_placed.SetActive(value: true);
                    intText.SetActive(value: false);
                    open.Play();
                    opened = true;
                }
        }
        if (locked)
        {
                lockedtext.SetActive(value: true);
        }
}

private void OnTriggerExit(Collider other)
{
        if (other.CompareTag("MainCamera"))
        {
                intText.SetActive(value: false);
                lockedtext.SetActive(value: false);
        }
}

private void Update()
{
        if (keyfound)
        {
                locked = false;
        }
}
}
```

SettingsButton.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SettingsButtons : MonoBehaviour
{
public GameObject PauseMenu, SettingsMenu;

public void Resume() //Resumes game and lets player move again
{
	Debug.Log("Resumed");
	PauseMenu.SetActive(false);
	SettingsMenu.SetActive(false);
	Time.timeScale = 1;
	Cursor.lockState = CursorLockMode.Locked;
	Cursor.visible = false;
}

public void sens()
{
}


public void brightness()
{
}

public void Back()
{
	PauseMenu.SetActive(false);
	SettingsMenu.SetActive(true);
}
}
```

SettingsScript.cs

```
using UnityEngine;
using UnityStandardAssets.Characters.FirstPerson;


public class SettingsScript : MonoBehaviour
{
    public GameObject PauseMenu, SettingsMenu;
    public FirstPersonControllerCustom playerController;
    public GameObject CameraOverlay, StaminaCanvas;


    void Start()
    {

    }

    void Update() //If Q is pressed, closes options menu
    {
        if (Input.GetKeyDown(KeyCode.Q))
        {
            if (PauseMenu.activeSelf)
            {
                Resume();
                CameraOverlay.SetActive(true);
                StaminaCanvas.SetActive(true);
            }
        }

    }


    public void Resume()
    {
        PauseMenu.SetActive(false);
        SettingsMenu.SetActive(false);
        Time.timeScale = 1;
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
```

```
            playerController.canMove = true;
            CameraOverlay.SetActive(true);
            StaminaCanvas.SetActive(true);
        }

    }
```

SmileyScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SmileyScript : MonoBehaviour
{

    public Rigidbody monsRigid;
    public Transform monsTrans;
    public Transform playTrans;

    public int monsSpeed;

    public float detectionRange = 10f;


    private bool isPlayerLooking = false;
    private float lookTimer = 0f;
    private float teleportCooldown = 2f;
    private Vector3 teleportDestination;


    private void FixedUpdate()
    {
        monsRigid.velocity = transform.forward * monsSpeed * Time.deltaTime;
    }

    private void Update()
    {
        monsTrans.LookAt(playTrans);


        float distanceToPlayer = Vector3.Distance(monsTrans.position,
playTrans.position);
        if (distanceToPlayer <= detectionRange)
        {

            if (LookingAtMonster())
```

```
            {
                lookTimer += Time.deltaTime;
                if (lookTimer >= teleportCooldown)
                {
                    TeleportMonster();
                    lookTimer = 0f;
                }
            }
            else
            {
                lookTimer = 0f;
            }
        }
        else
        {
            lookTimer = 0f;
        }
    }

    private bool LookingAtMonster()
    {
        Vector3 directionToMonster = monsTrans.position - playTrans.position;
        float angle = Vector3.Angle(playTrans.forward, directionToMonster);
        return angle < 45f;
    }

    private void TeleportMonster()
    {
        float randomX = Random.Range(-10f, 10f);
        float randomZ = Random.Range(-10f, 10f);
        teleportDestination = new Vector3(randomX, 0f, randomZ);

        monsTrans.position = teleportDestination;
    }
}
```

StaminaController.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityStandardAssets.Characters.FirstPerson;

public class StaminaController : MonoBehaviour
{
[Header("Stamina Main Parameters")]
public float playerStamina = 100f;

[SerializeField]
private float maxStamina = 100f;

[SerializeField]
private float jumpCost = 20f;

[HideInInspector]
public bool hasRegenerated = true;

[HideInInspector]
public bool weAreSprinting;

[Header("Stamina Regen Parameters")]
[Range(0f, 50f)]
[SerializeField]
private float staminaDrain = 0.5f;

[Range(0f, 50f)]
[SerializeField]
private float staminaRegen = 0.5f;

[Header("Stamina Speed Parameters")]
[SerializeField]
private int slowedRunSpeed = 4;

[SerializeField]
private int normalRunSpeed = 8;
```

```
[Header("Stamina UI Elements")]
[SerializeField]
private Image staminaProgressUI;

[SerializeField]
private CanvasGroup sliderCanvasGroup;

private FirstPersonControllerCustom playerController;

private void Start()
{
	playerController = GetComponent<FirstPersonControllerCustom>();
}

private void Update()
{
	if (!weAreSprinting && (double)playerStamina <= (double)maxStamina - 0.01)
	{
		playerStamina += staminaRegen * Time.deltaTime;
		UpdateStamina(1);
		if (playerStamina >= maxStamina)
		{
			playerController.SetRunSpeed(normalRunSpeed);
			sliderCanvasGroup.alpha = 0f;
			hasRegenerated = true;
		}
	}
}

public void Sprinting()
{
	if (hasRegenerated)
	{
		weAreSprinting = true;
		playerStamina -= staminaDrain * Time.deltaTime;
		UpdateStamina(1);
		if (playerStamina == 0f)
		{
			hasRegenerated = false;
			sliderCanvasGroup.alpha = 0f;
		}
	}
	if (playerStamina <= 20f)
```

```
        {
                playerController.SetRunSpeed(slowedRunSpeed);
        }
        else
        {
                playerController.SetRunSpeed(normalRunSpeed);
        }
}

private void UpdateStamina(int value)
{
        staminaProgressUI.fillAmount = playerStamina / maxStamina;
        if (value == 0)
        {
                sliderCanvasGroup.alpha = 0f;
        }
        else
        {
                sliderCanvasGroup.alpha = 1f;
        }
}
}
```

Timer.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class Timer : MonoBehaviour
{
    [SerializeField] TextMeshProUGUI timerText;
    float elapsedTime;

    void Update()
    {
        elapsedTime += Time.deltaTime;
        int minutes = Mathf.FloorToInt(elapsedTime / 60);
        int seconds = Mathf.FloorToInt(elapsedTime % 60);
        timerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);
    }

    void OnDestroy()
    {
        PlayerPrefs.SetFloat("ElapsedTime", elapsedTime);
    }
}
```

VolumeScript.cs

```
using UnityEngine;
using UnityEngine.UI;

public class VolumeScript : MonoBehaviour
{
    public Slider volumeSlider;

    private void Start()
    {
        volumeSlider.value = AudioListener.volume;
    }

    public void SetVolume(float volume)
    {
        AudioListener.volume = volume;
    }
}
```