SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5503

ALGORITAM ODREĐIVANJA PORTALA U PROIZVOLJNOJ SCENI

Dino Ehman

Zagreb, Lipanj, 2018

Sadržaj

1. Uvod

Videoigre su elektroničke igre koje uključuju interakciju s korisničkim sučeljem kako bi generirali vizualne povratne informacije na video uređajima poput televizora ili zaslona računala. Povijest videoigara seže do 1947. godine kada je Thomas T. Goldsmith dobio ideju videoigre i patentirao je [1]. Videoigre se razvijaju iz dana u dan te im se poboljšavaju tehničke karakteristike. U današnje vrijeme industrija videoigara jedna je od najprofitabilnijih industrija, što dokazuje činjenica da je prodaja videoigara u 2016. godini ostvarila 500 milijuna dolara profita, dok je predviđanje da će 2019. godine premašiti 1 milijardu dolara. Jedan od koncepata videoigara su i portali te iscrtavanje portala (eng. portal rendering). U računalnoj grafici iscrtavanje portala je način smanjenja broja poligona i time optimiziranje brzine crtanja scene. Iscrtavač portala (eng. portal renderer) je iscrtavač koji enkapsulira dijelove svijeta igara (razine) u poligonalnim tijelima. Svaka ravnina takvog poligonalnog tijela može voditi do drugog takvog tijela, čineći portal. Na primjer, u videoigri, područje igre može biti podijeljeno na više sektora. Ti sektori bi tada bili međusobno povezani otvorima kao što su vrata ili prozori. Ti se otvori nazivaju portalima. Kada se sektor iza portala treba iscrtati, jedini vidljivi dijelovi su dijelovi koji se mogu vidjeti kroz portal. Stoga se sektor može odrezati na granice portala kako bi se smanjilo nepotrebno iscrtavanje [2]. U ovome radu će biti prikazano kako napraviti portale u scenama te kako napraviti da prikaz unutrašnjosti okvira portala je zapravo pogled iz njemu uparenog portala. Također, bit će pokazano kako ostvariti rekurzivno iscrtavanje portala ako se upareni portali nađu jedan ispred drugoga.

2. Općenito o portalima

U arhitekturi, portal je otvor u zidu zgrade, vrata ili utvrde, posebice veliki ulaz u važnu građevinu. Portali, od vremena drevnih Egipatskih pilona (kule pred ulazom u hram) i Babilonskih gradskih vrata su postali spomenici sami za sebe, naglašavajući pojačanu značajnost onoga što slijedi iza njih. Na slici 1. možemo vidjeti jedan primjer tih portala.

Slika 1. Portal katedrale Chartres u Francuskoj

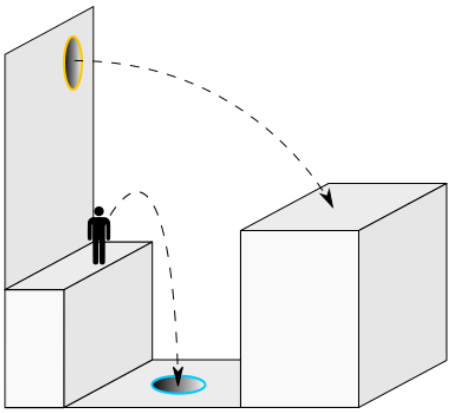
U znanstvenoj fantastici i fikciji, riječ portal pak ima drugačije značenje. Portal predstavlja tehnološka ili magična vrata koja spajaju dvije udaljene lokacije razdvojene kroz vrijeme i prostor. Obično se sastoji od dva ili više prolaza, gdje objekt ulazeći u jedan prolaz trenutačno izlazi iz drugog prolaza. Mjesto koja su povezana portalima uključuju drugačije mjesto u istom svemiru (u tom slučaju su

alternativa teleportaciji), paralelne svjetove (interdimenzionalni portali), prošlost ili budućnost (vremenski portal) i druge razine postojanja.

Slika 2. Primjer portala

2.1 Portali u video igrama

Portali, poznati također kao warp zone ili teleporteri su elementi u dizajnu video igara koji omogućuju igračima trenutačno putovanje između dvije lokacije ili razine. Na slici 3. možemo vidjeti kako prolazeći kroz plavi portal s neke visine stvara momentum prilikom izlaska kroz narančasti portal [3].

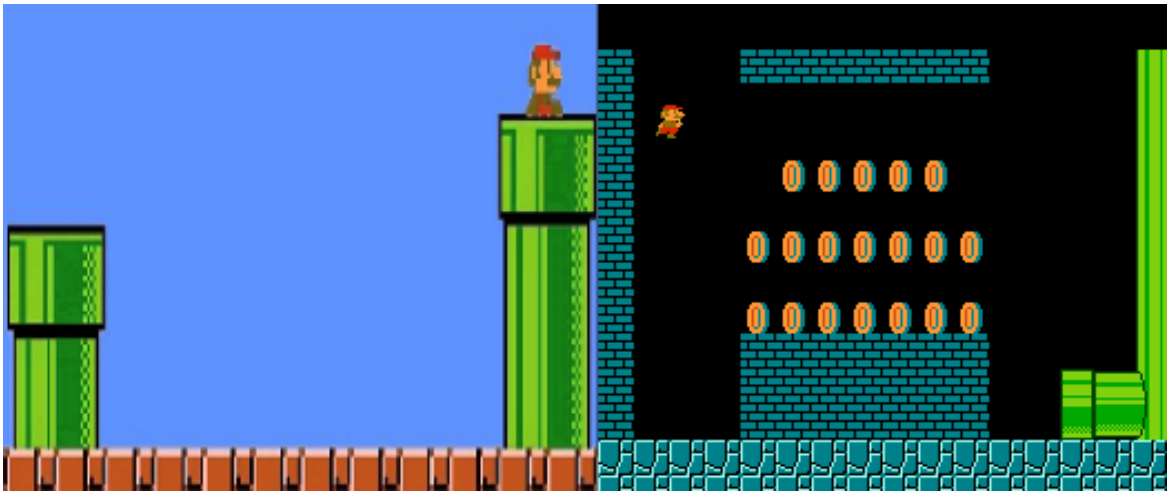


Slika 3. Osnovni koncept portala

Portali mogu biti tajni prolazi, dostupni jedino igračima koji su sposobni ih pronaći, iako se najčešće koriste kao glavno sredstvo putovanja u određenim igrama. Portali mogu namjerno biti postavljeni unutar zagonetki, mogu biti korišteni da se izbjegnu

ozljede u dijelovima igre koji su već prije bili završeni, mogu biti nešto što igrač može iskoristiti za varanje ili biti korišteni kao kazna ako igrači skrenu s ispravnog puta.

U Super Mario franšizi, igrači mogu proći kroz određene cijevi koje će ih odvesti u skrivene sobe ili dodatne razine (slika 4.)



Slika 4. Mario prolazi kroz cijev koja vodi u tajnu sobu

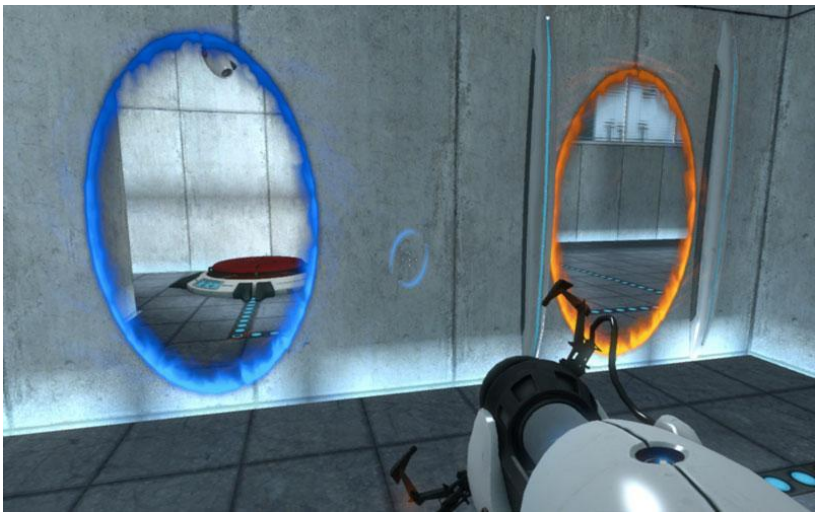
U Spyro franšizi, svaki zasebna razina ili svijet su razdvojeni portalima što je dizajnerima omogućilo zaslone za učitavanje koji ne prekidaju kontinuitet igre (slika 5.)

Slika 5. Spyro ulazi kroz portal do druge razine

2.2 Portali u Portal franšizi

Video igra Portal i njezin nastavak Portal 2, kreirani od strane tvrtke Valve, kao glavnu značajku igre koriste uređaj koji je sposoban kreirati portale, poznatiji kao *Aperture Science Handheld Portal Device* (slika 6). Uređaj služi kao centralna mehanika igre za rješavanje raznih zagonetki i pristup inače nedostupnim mjestima.

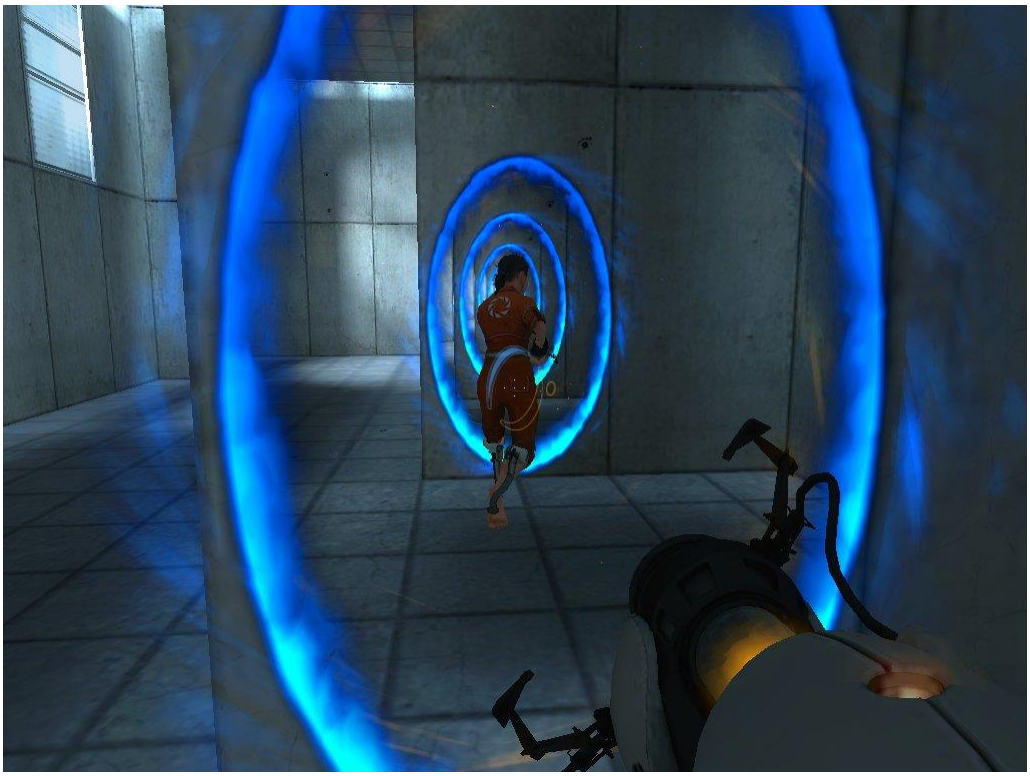
Slika 6. Portal uređaj



Slika 7. Prikaz portala u igrici Portal

Na slici 7. možemo vidjeti prikaz portala u igrici Portal. Portali su prikazani s par specijalnih efekata, vanjski dio portala je obrubljen s plavim i narančastim česticama

dok unutrašnjost prikazuje što drugi portal „vidi”, tj. prikazuje mjesto gdje bi igrač došao kada bi prošao kroz portal. Ukoliko bi postavili portale tako da izlaz jednog portala gleda u ulaz drugog portala, možemo vidjeti efekt rekurzivnog prikaza portala (slika 8). Taj specifičan efekt će biti objašnjen u radu te kako ga implementirati.



Slika 8. Rekurzivni prikaz portala

3. Portali u OpenGL-u

Želimo implementirati teleportacijsku funkciju, gdje su izvor i odredište prikazani kao rupe u zidovima kroz koje se može vidjeti. Portal koji se iscrtava ćemo nazvati izvorišni portal, a portal koji je povezan s njim odredišni portal. Prolazak kroz izvorišni portal i izlazak na odredišnom portalu treba biti trenutačno i bez grubih prijelaza. Također je moguće postaviti portale licem u lice kako bi se kreirao rekurzivni efekt, tj. beskonačna dubina.

3.1 OpenGL

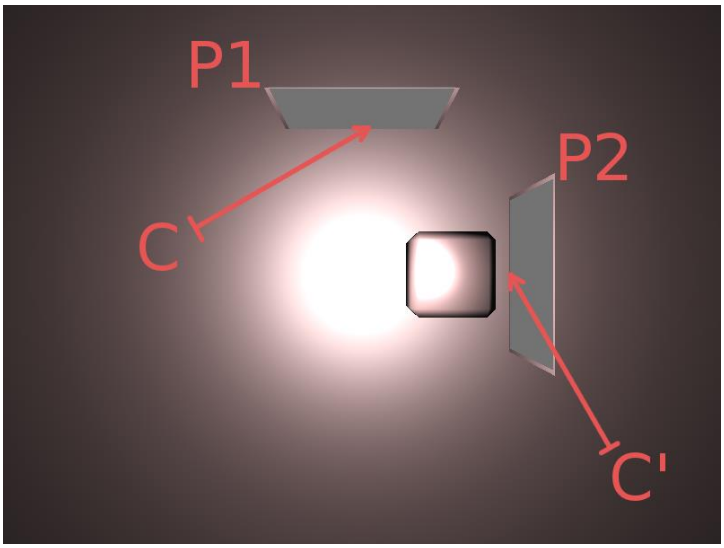
OpenGL ili Javna Grafička Biblioteka (eng. Open Graphics Library) je grafički 3D standard stvoren od strane Silicon Graphics tvrtke. Standard se sastoji od preko 250 različitih funkcija koje definiraju načine crtanja kompleksnih 3D scena s jednostavnim primitivima ili osnovnim geometrijskim elementima kao npr. kocka, trokut, piramida. OpenGL je popularna biblioteka u industriji video igara uz Microsoftov Direct3D. Također koristi se u područjima virtualne realnosti, dizajna potpomognutim računalom (eng. computer-aided design, CAD), znanstvenih vizualizacija, informacijskih vizualizacija, simulacija leta te brojnih drugih.

3.2 Iscrtavanje jednog portala (bez rekurzije)

Prilikom iscrtavanja scene gdje je portal unutar pogleda kamere, sve će biti potrebno iscrtati dva puta, prvo se iscrta scena izvan okvira portala i zatim unutar okvira portala. Scena koja se iscrtava unutar okvira portala je nacrtana iz perspektive virtualne kamere (slika 9). Položaj i orijentacija virtualne kamere je izračunata na sljedeći način. Uzme se udaljenost i kut (transformacijska matrica) od kamere do izvorišnog portala i inverz te matrice se primijeni na odredišni portal kako bi dobili položaj i kut virtualne kamere. Izračun položaja virtualne kamere možemo vidjeti kroz sljedeći kôd [3].

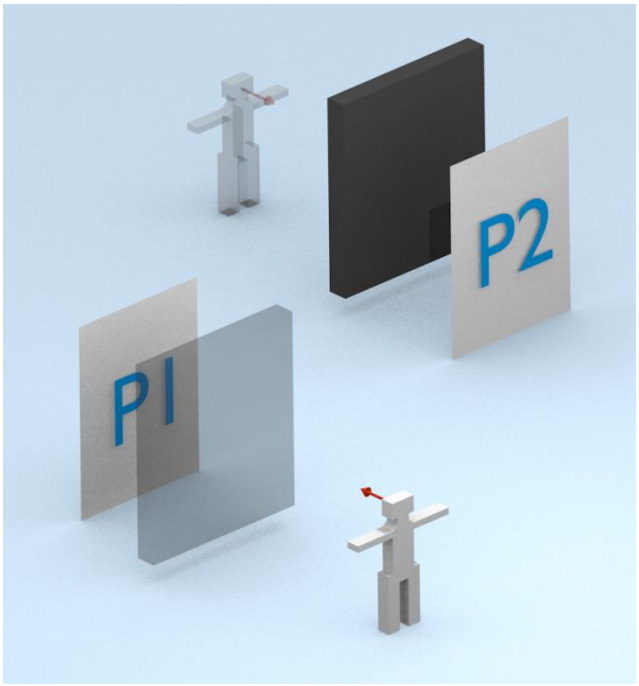
```
/**
 * Compute a world2camera view matrix to see from portal 'dst', given
 * the original view and the 'src' portal position.
 */
glm::mat4 portal_view(glm::mat4 orig_view, Mesh* src, Mesh* dst) {
  glm::mat4 mv = orig_view * src->object2world;
  glm::mat4 portal_cam =
    // 3. transformation from source portal to the camera - it's the
    //    first portal's ModelView matrix:
    mv
    // 2. object is front-facing, the camera is facing the other way:
    * glm::rotate(glm::mat4(1.0), glm::radians(180.0f),
glm::vec3(0.0,1.0,0.0))
    // 1. go the destination portal; using inverse, because camera
    //    transformations are reversed compared to object
    //    transformations:
    * glm::inverse(dst->object2world)
    ;
  return portal_cam;
}
```

Isječak programskog kôda 1. Izračun položaja i matrice pogleda virtualne kamere



Slika 9. Virtualna kamera C' koju želimo dobiti iz kamere C

Ovaj pristup radi dobro ali postoji jedan problem, ne smije postojati objekt iza odredišnog portala, tj. između virtualne kamere i odredišnog portala. Ako bi postojao takav objekt on bi spriječio naš pogled virtualne scene unutar portala (slika 10).



Slika 10. Crni objekt iza odredišnog portala P2 sprječava pogled

Rješenje ovog problema je promijeniti matricu pogleda virtualne kamere tako da bliska odrezajuća ploha počinje na plohi odredišnog portala. Ova tehnika se zove *oblique view frustum near-plane clipping* [4][5]. Implementaciju ove tehnike možemo vidjeti u isječku kôda 2 [4][5].

```
inline float sgn(float a)
{
    if (a > 0.0F) return (1.0F);
    if (a < 0.0F) return (-1.0F);
    return (0.0F);
}
void ModifyProjectionMatrix(const Vector4D& clipPlane)
{
    float matrix[16];
    Vector4D q;
    // Grab the current projection matrix from OpenGL
    glGetFloatv(GL_PROJECTION_MATRIX, matrix);

    // Calculate the clip-space corner point opposite the clipping plane
    // as (sgn(clipPlane.x), sgn(clipPlane.y), 1, 1) and
    // transform it into camera space by multiplying it
    // by the inverse of the projection matrix
    q.x = (sgn(clipPlane.x) + matrix[8]) / matrix[0];
    q.y = (sgn(clipPlane.y) + matrix[9]) / matrix[5];
    q.z = -1.0F;
    q.w = (1.0F + matrix[10]) / matrix[14];

    // Calculate the scaled plane vector
    Vector4D c = clipPlane * (2.0F / Dot(clipPlane, q));

    // Replace the third row of the projection matrix
    matrix[2] = c.x;
    matrix[6] = c.y;
    matrix[10] = c.z + 1.0F;
    matrix[14] = c.w;
    // Load it back into OpenGL
    glMatrixMode(GL_PROJECTION);
    glLoadMatrix(matrix);
}
```

Isječak programskog kôda 2. Promjena matrice pogleda virtualne kamere

Nakon što imamo točnu virtualnu matricu pogleda, možemo stvarno iscrtati portal. Postoje dvije popularne tehnike za napraviti to, iscrtavanje na teksturu (eng. rendering to a texture) i korištenje spremnika maske (eng. stencil bufffer).

Tehnika iscrtavanja na teksturu je obično ostvarena u OpenGL-u koristeći FrameBuffer objekte (FBO). Proces iscrtavanja je sljedeći:

1. Stvoriti FBO i postaviti ga kao trenutačni cilj iscrtavanja
2. Generirati matricu pogleda virtualne kamere koristeći metodu *oblique view frustum near-plane clipping*
3. Iscrtati scenu iz pogleda virtualne kamere i popuniti FBO s iscrtanim okvirom
4. Iscrtati scenu normalno iz pogleda kamere ali primijeniti novo generiranu teksturu FBO-a na okvir portala

Postoji jedan veliki nedostatak ove tehnike, a to je rezolucija teksture. Gledano iz dalje udaljenosti, ova tehnika bi izgledala dobro, no kada bi se dovoljno približili portalu pojedinačni pikseli teksture bi se počeli jasno viđati. Potencijalni popravak bi bilo povećati veličinu teksture no to bi utjecalo na pojačano korištenje GPU memorije, pogotovo kod rekurzivnog pristupa.

Druga način iscrtavanja portala je korištenje spremnika maske (eng. stencil buffer). To je dodatni spremnik u grafičkoj kartici (uz spremnik boje (eng. color buffer) i spremnik udaljenosti od očišta (eng. depth buffer)) koji se koristi kako bi se zamaskirali određeni dijelovi zaslona za sprječavanje ili dozvolu crtanja na razini piksela. To znači da za svaki piksel na zaslonu postoji broj (obično 8 bita) u spremniku maske koji GPU može pogledati da odredi da li treba iscrtati piksel ili ne. To je pogodno zbog mnogostrukih prolazaka iscrtavanja s različitim sadržajem spremnika maski i možemo iskoristiti to kako bi iscrtali i unutrašnjost i vanjski dio okvira portala bez korištenja odvojenih ciljeva iscrtavanja. Prilikom korištenja spremnika maske trebamo razmišljati o dvije stvari, želimo li gledati sadržaj spremnika maske kako bi odredili treba li nacrtati piksel ili ne i želimo li promijeniti sadržaj spremnika maske prilikom crtanja te ako da, ovisno o kojem uvjetu.

Prva situacija se može regulirati omogućavanjem, odnosno onemogućavanjem testa maske, sa `glEnable(GL_STENCIL_TEST)` ili `glDisable(GL_STENCIL_TEST)`.

Prilikom crtanja piksela s omogućenim testom maske grafička kartica će primijeniti test između trenutačne vrijednosti spremnika maske i referentne vrijednosti. Oba parametra, tip testa i referentna vrijednost, se mogu postaviti s funkcijom glStencilFunc.

Druga situacija se može regulirati s operacijom maske. Ona se postavlja s funkcijom glStencilOp koji zahtijeva tri argumenta:

- sfail – operacija koja se izvršava ako test maske padne
- dpfail – operacija koja se izvršava ako test maske prođe, ali test udaljenosti padne
- dppass – operacija koja se izvršava ako i test maske i test udaljenosti prođu

Sva tri argumenta primaju operacije poput:

- GL_KEEP – ostavlja trenutačnu vrijednost maske
- GL_INCR – povećava vrijednost maske za jedan
- GL_DECR – smanjuje vrijednost maske za jedan

Proces koji koristimo u iscrtavanju portala pomoću spremnika maske je sljedeći:

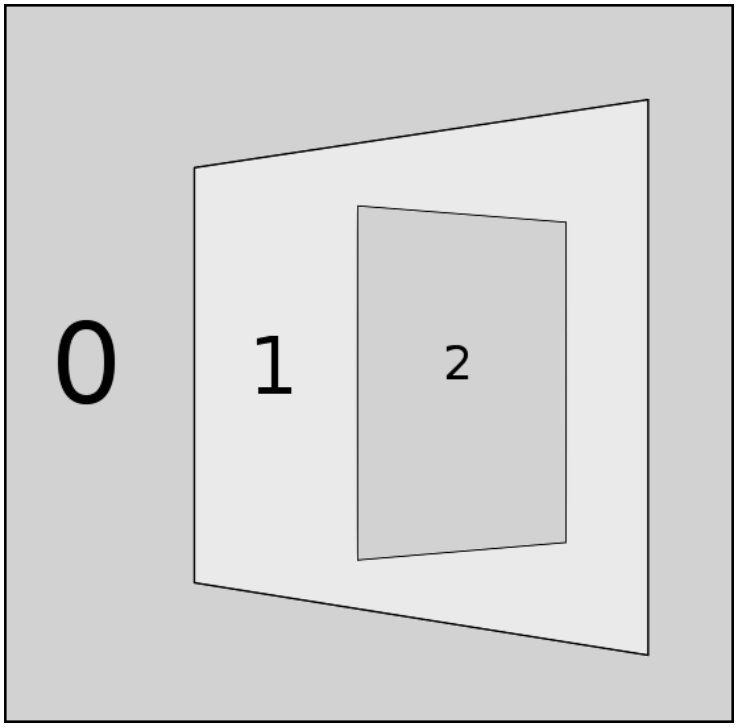
1. Onemogućiti pisanje u spremnik boje i spremnik udaljenosti od očišta, ali omogućiti pisanje u spremnik boje
2. Postaviti operaciju maske GL_INCR na stfail, što znači da će se vrijednost maske povećati za jedan kada test maske padne
3. Postaviti funkciju maske na GL_NEVER, koja osigurava da test maske uvijek padne na svakom pikselu koji treba nacrtati
4. Iscrtamo okvir portala. Nakon iscrtavanja spremnik maske je napunjen s nulama izvan okvira portala i s jedinicama unutar okvira portala
5. Generirati matricu pogleda virtualne kamere
6. Onemogućiti pisanje u spremnik maske ali omogućiti pisanje u spremnik boje i spremnik udaljenosti od očišta
7. Postaviti funkciju maske GL_EQUAL s referentom vrijednosti 1. Sada će se samo iscrtati pikseli gdje je vrijednost maske jednaka 1, što je unutrašnjost okvira portala
8. Iscrtati scenu koristeći virtualnu kameru iz koraka 5.

9. Onemogućiti test maske i pisanje u spremnik boje ali omogućiti pisanje u spremnik udaljenosti od očišta
10. Očistiti spremnik udaljenosti od očišta
11. Iscrtati okvir portala još jednom, ovaj puta s očišćenim spremnikom udaljenosti od očišta
12. Omogućiti spremnik boja
13. Iscrtati cijelu scenu s glavnom kamerom

Razlog zašto je okvir portala iscrtan dva puta, jednom sa spremnikom maske i jednom sa spremnikom udaljenosti od očišta je zbog toga što želimo da portal bude zaklonjen ako postoji stvarni objekt ispred njega u sceni.

3.3 Iscrtavanje rekurzivnih portala

Osnovna tehnika prilikom korištenja metode spremnika maske rekurzivno je povećavanje vrijednosti maske za svaku razinu dubine. Tako možemo iscrtati n-tu virtualnu scenu u području koje ima vrijednost maske n (slika 11.)



Slika 11. Rekurzivni portali

Ovaj pristup može biti postignut s rekurzivnim popunjavanjem spremnika maske s bilo kojim okvirom portala na trenutnoj razini rekurzije, zatim je potrebno povećati vrijednost maske s iscrtavanjem virtualnih portala na sljedećoj razini s GL_INCR operacijom. Kada dosegnemo maksimalnu dubinu rekurzije, počinjemo iscrtavati virtualne scene od vrha prema dnu, počevši s najvećom vrijednosti maske, sve dok nismo nazad na nuli. Implementaciju ove metode možemo vidjeti u isječku kôda 3 [6].

```
void Scene::drawRecursivePortals(glm::mat4 const &viewMat, glm::mat4 const
&projMat, size_t maxRecursionLevel, size_t recursionLevel)
{
    for (auto &pair : d_portals)
    {
        Portal &portal = *pair.second;
        // Disable color and depth drawing
        glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
        glDepthMask(GL_FALSE);

        // Disable depth test
        glDisable(GL_DEPTH_TEST);

        // Enable stencil test, to prevent drawing outside
        // region of current portal depth
        glEnable(GL_STENCIL_TEST);

        // Fail stencil test when inside of outer portal
        // (fail where we should be drawing the inner portal)
        glStencilFunc(GL_NOTEQUAL, recursionLevel, 0xFF);

        // Increment stencil value on stencil fail
        // (on area of inner portal)
        glStencilOp(GL_INCR, GL_KEEP, GL_KEEP);

        // Enable (writing into) all stencil bits
        glStencilMask(0xFF);
```

```
        // Draw portal into stencil buffer
        portal.draw(viewMat, projMat);

        // Calculate view matrix as if the player was already teleported
        glm::mat4 destView = viewMat * portal.modelMat()
            * glm::rotate(glm::mat4(1.0f), 180.0f, glm::vec3(0.0f, 1.0f, 0.0f)
* portal.orientation())
            * glm::inverse(portal.destination()->modelMat());

        // Base case, render inside of inner portal
        if (recursionLevel == maxRecursionLevel)
        {
            // Enable color and depth drawing
            glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
            glDepthMask(GL_TRUE);

            // Clear the depth buffer so we don't interfere with stuff
            // outside of this inner portal
            glClear(GL_DEPTH_BUFFER_BIT);

            // Enable the depth test
            // So the stuff we render here is rendered correctly
            glEnable(GL_DEPTH_TEST);

            // Enable stencil test
            // So we can limit drawing inside of the inner portal
            glEnable(GL_STENCIL_TEST);

            // Disable drawing into stencil buffer
            glStencilMask(0x00);

            // Draw only where stencil value == recursionLevel + 1
            // which is where we just drew the new portal
            glStencilFunc(GL_EQUAL, recursionLevel + 1, 0xFF);
```

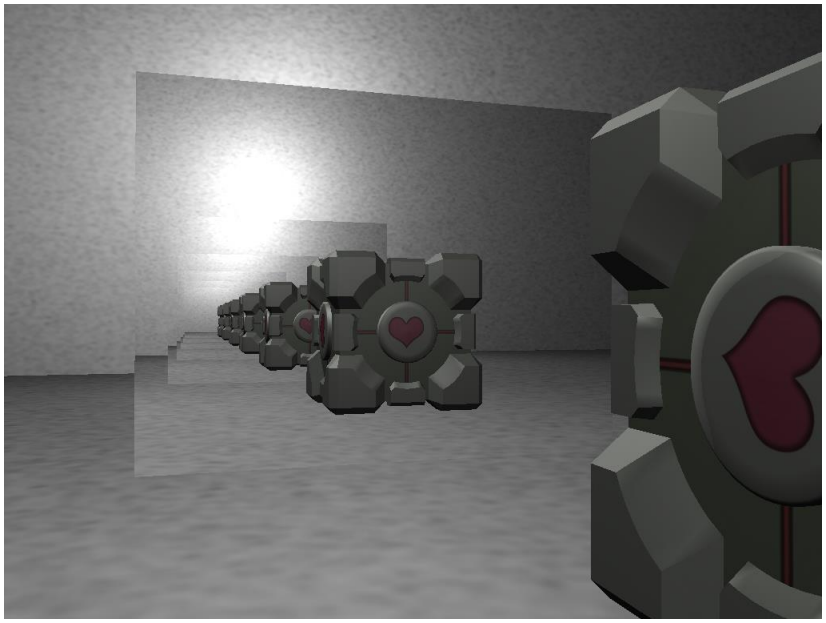
```
            // Draw scene objects with destView, limited to stencil buffer
            // use an edited projection matrix to set the near plane to the
portal plane
            drawNonPortals(destView, portal.clippedProjMat(destView,
projMat));
            //drawNonPortals(destView, projMat);
        }
        else
        {
            // Recursion case
            // Pass our new view matrix and the clipped projection matrix (see
above)
            drawRecursivePortals(destView, portal.clippedProjMat(destView,
projMat), maxRecursionLevel, recursionLevel + 1);
        }
        // Disable color and depth drawing
        glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
        glDepthMask(GL_FALSE);

        // Enable stencil test and stencil drawing
        glEnable(GL_STENCIL_TEST);
        glStencilMask(0xFF);

        // Fail stencil test when inside of our newly rendered
        // inner portal
        glStencilFunc(GL_NOTEQUAL, recursionLevel + 1, 0xFF);

        // Decrement stencil value on stencil fail
        // This resets the incremented values to what they were before,
        // eventually ending up with a stencil buffer full of zero's again
        // after the last (outer) step.
        glStencilOp(GL_DECR, GL_KEEP, GL_KEEP);

        // Draw portal into stencil buffer
        portal.draw(viewMat, projMat);
    }
```

```
// Disable the stencil test and stencil writing
glDisable(GL_STENCIL_TEST);
glStencilMask(0x00);

// Disable color writing
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);

// Enable the depth test, and depth writing.
glEnable(GL_DEPTH_TEST);
glDepthMask(GL_TRUE);

// Make sure we always write the data into the buffer
glDepthFunc(GL_ALWAYS);

// Clear the depth buffer
glClear(GL_DEPTH_BUFFER_BIT);

// Draw portals into depth buffer
for (auto &pair : d_portals)
    pair.second->draw(viewMat, projMat);

// Reset the depth function to the default
glDepthFunc(GL_LESS);

// Enable stencil test and disable writing to stencil buffer
glEnable(GL_STENCIL_TEST);
glStencilMask(0x00);

// Draw at stencil >= recursionlevel
// which is at the current level or higher (more to the inside)
// This basically prevents drawing on the outside of this level.
glStencilFunc(GL_GEQUAL, recursionLevel, 0xFF);

// Enable color and depth drawing again
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glDepthMask(GL_TRUE);
```

```
    // And enable the depth test
    glEnable(GL_DEPTH_TEST);

    // Draw scene objects normally, only at recursionLevel
    drawNonPortals(viewMat, projMat);

}
```

Isječak programskog kôda 3. Implementacija rekurzivnih portala u OpenGL-u

Na slici 12. možemo vidjeti kako izgledaju rekurzivni portali nastali korištenjem spremnika maske.



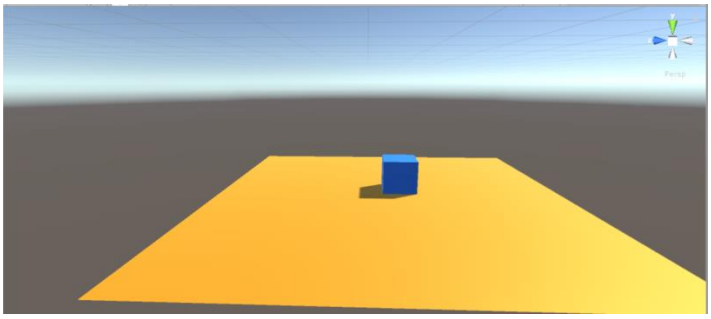
Slika 12. Rekurzivni portali nastali metodom spremnika maske

4. Portali u Unity-u

Što se tiče iscrtavanja portala u Unity-u, cilj nam je jednak kao i u OpenGL-u. Želimo napraviti portale koji u unutrašnjosti svojih okvira prikazuju pogled iz odredišnog portala. Također, želimo omogućiti mogućnost rekurzivnog prikaza portala ako se iz pogleda odredišnog portala ponovno vidi izvorišni portal te teleportacijsku funkciju koja će igrača prebaciti s pozicije izvorišnog portala na poziciju odredišnog portala.

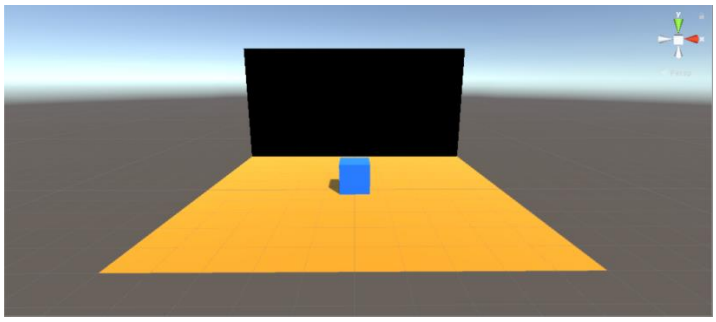
4.1 Iscrtavanje na teksturu

Isto kao u OpenGL-u, Unity također ima podršku za iscrtavanje na teksturu. Postupak kreiranja portala pomoću iscrtavanja na teksturu sličan je već prethodno opisanom postupku, no ipak je jednostavniji zbog grafičkog sučelja i već ugrađenih objekata. Na slici 13. imamo scenu s jednom ravninom i jednom kockom u koju ćemo ubaciti portal koristeći iscrtavanje na teksturu.



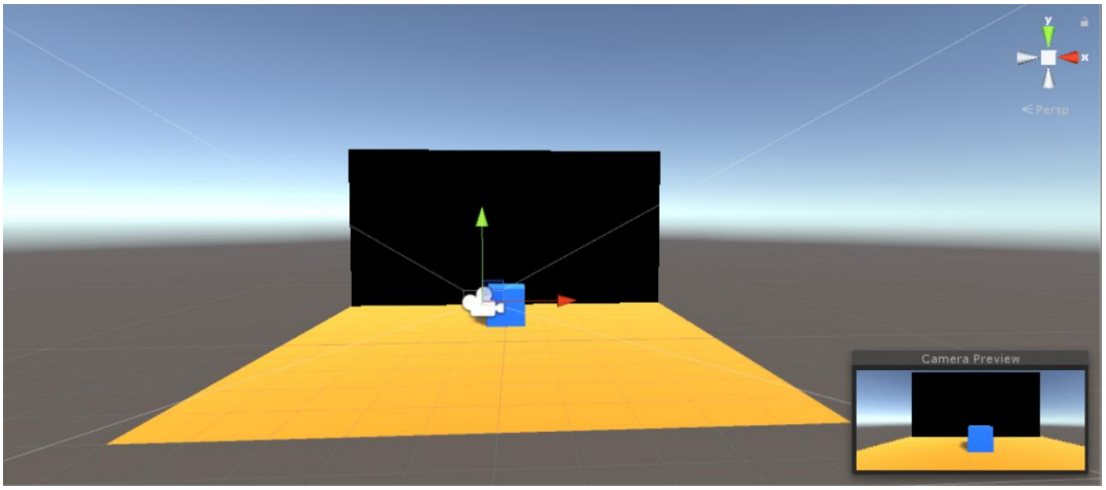
Slika 13. Nova scena

Prva stvar koju ćemo napraviti je dodati jedan novi četverokut u scenu (slike 14).



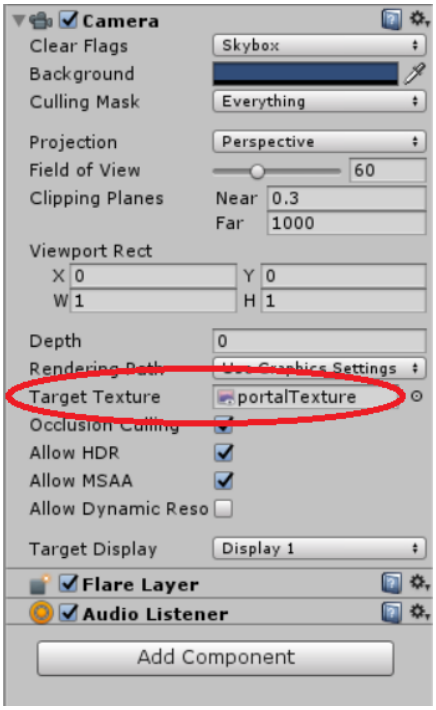
Slika 14. Prikaz četverokuta u sceni

Sljedeći korak je dodati novi Render Texture koji ćemo pridijeliti četverokutu i jednu novu kameru u scenu (slika 15).



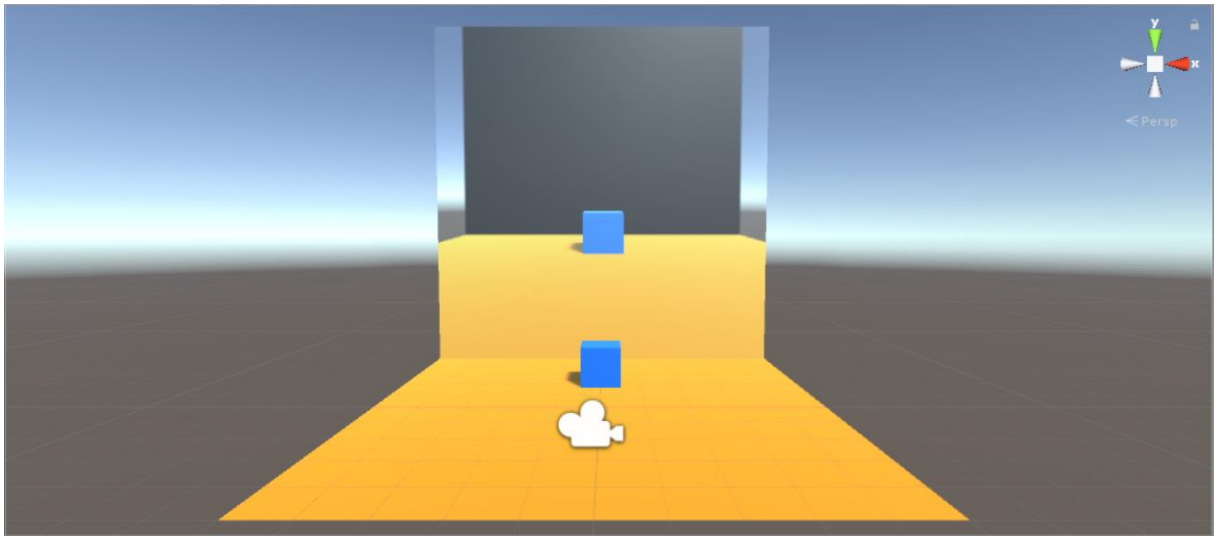
Slika 15. Nova kamera u sceni

Kameri u komponentu *Target Texture* potrebno je dodati napravljeni *Render Texture.*

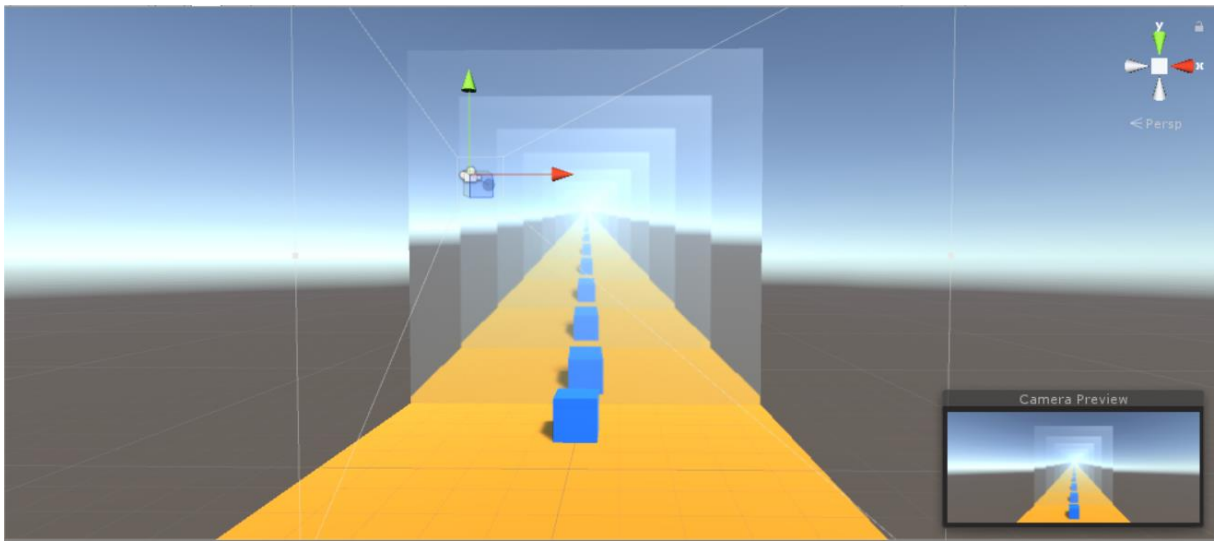


Slika 16. Target Texture u kameri

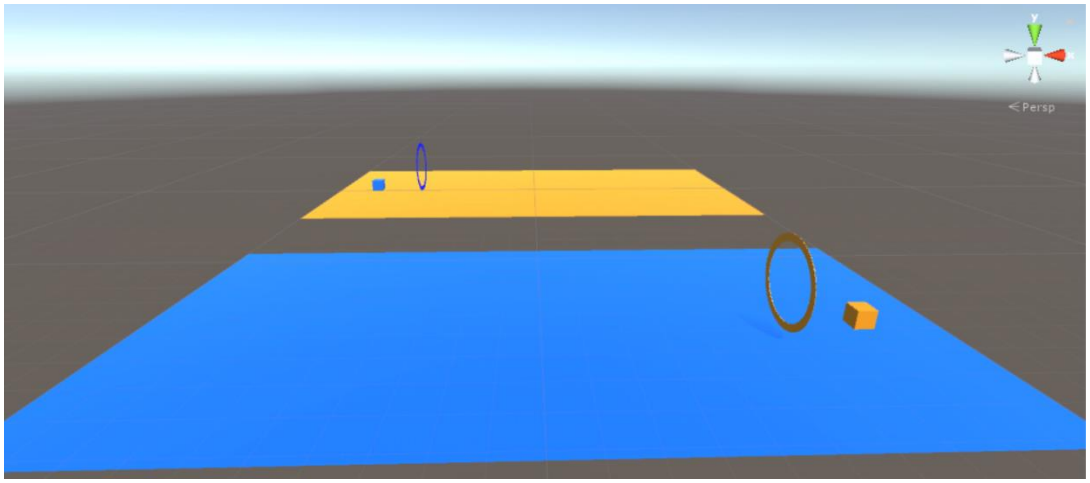
Rezultat pridjeljivanja teksture u komponentu kamere je vidljiv na slici 17. Crni kvadrat koji možemo vidjeti iza kocke je zato što je tekstura okrenuta licem u lice prema kameri i trenutačno je podržana samo jedna dubina rekurzije. Mijenjanjem postavki u teksturi dobivamo više rekurzivnih dubina te to možemo zatim primijeniti na portale (slika 18).



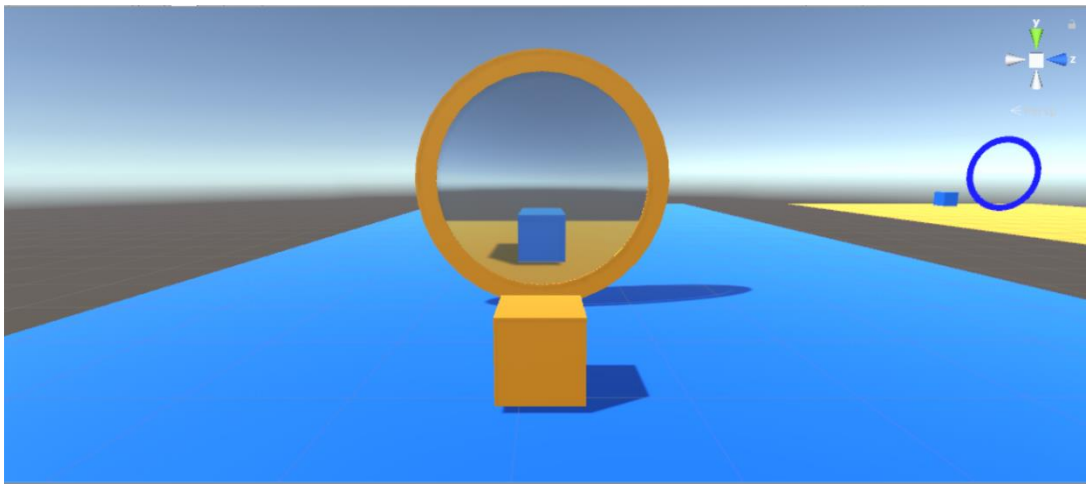
Slika 17. Tekstura bez rekurzije



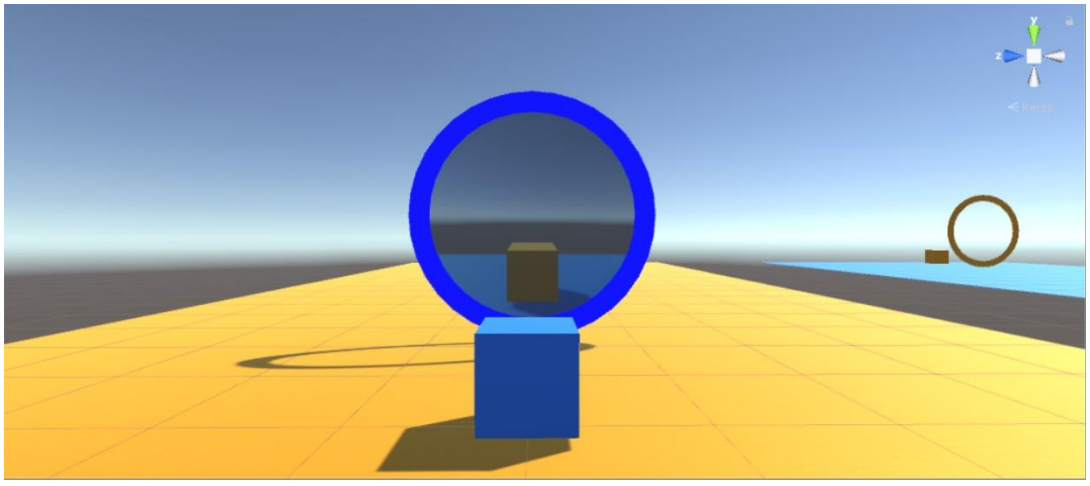
Slika 18. Tekstura uz rekurziju



Slika 19. Dva portala u sceni

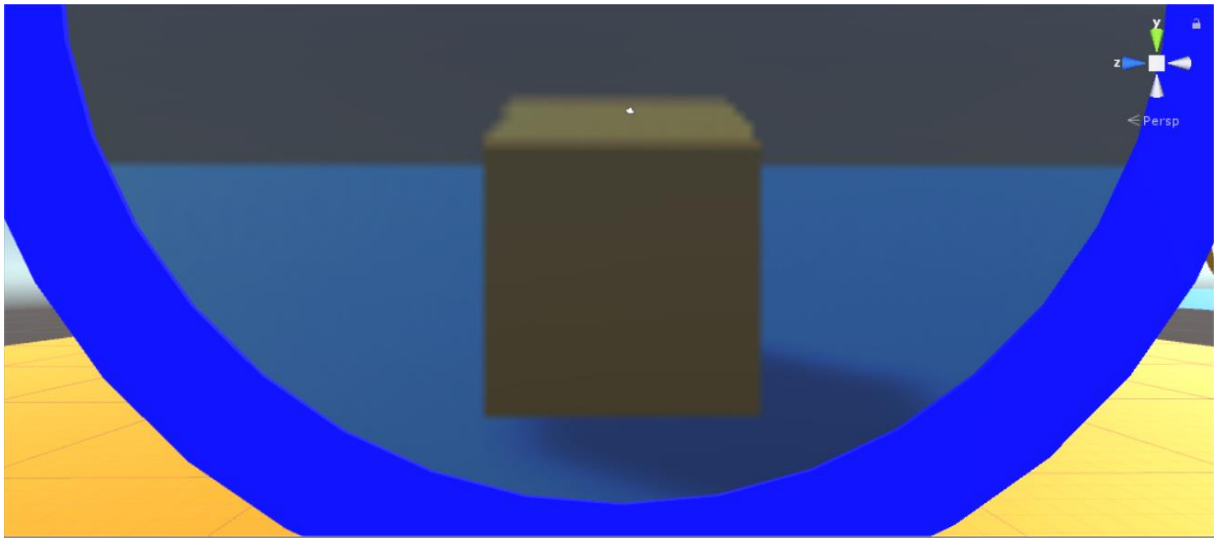


Slika 20. Pogled iz žutog portala

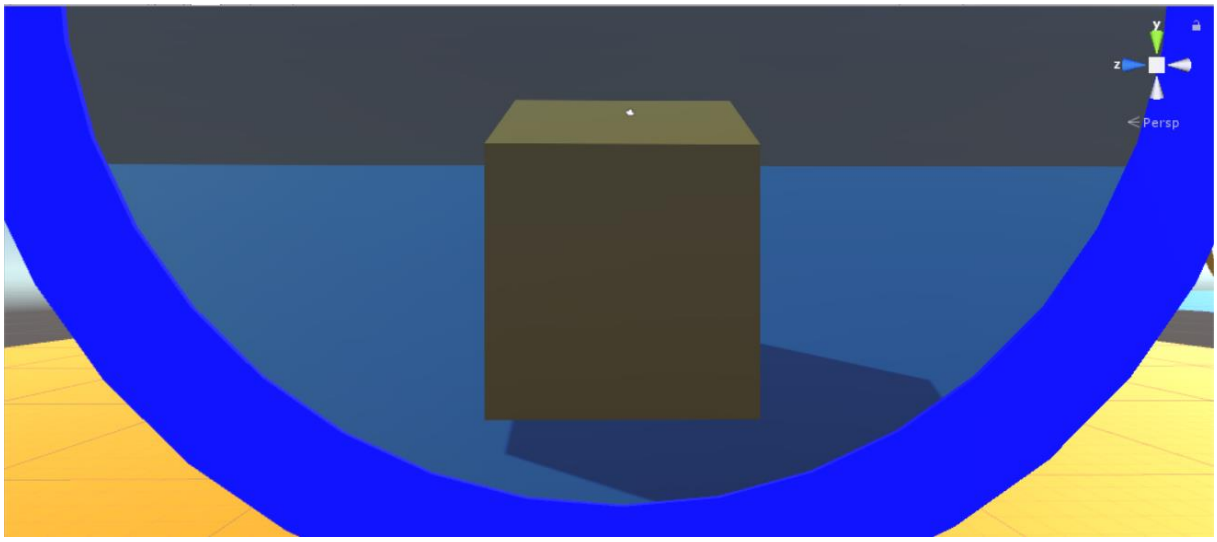


Slika 21. Pogled iz plavog portala

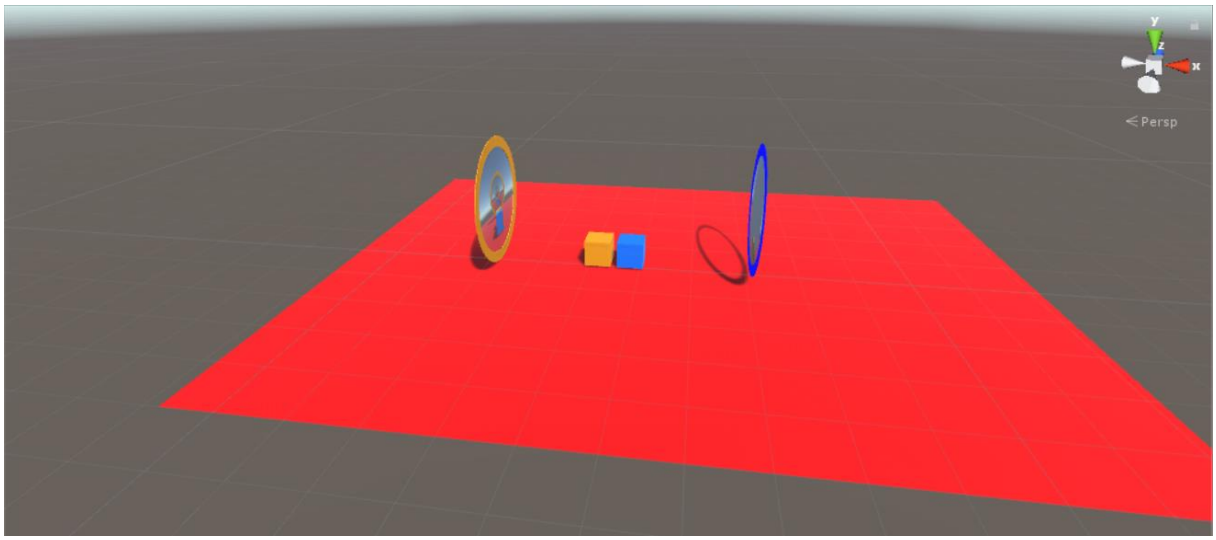
Već prije opisani problem iscrtavanja na teksturu (pikselizacija slike kada se dovoljno približimo portalu) možemo vidjeti na slici 22. Problem možemo riješiti s povećanjem rezolucije teksture (slika 23.), no time povećavamo potrošnju GPU memorije i usporavamo program.



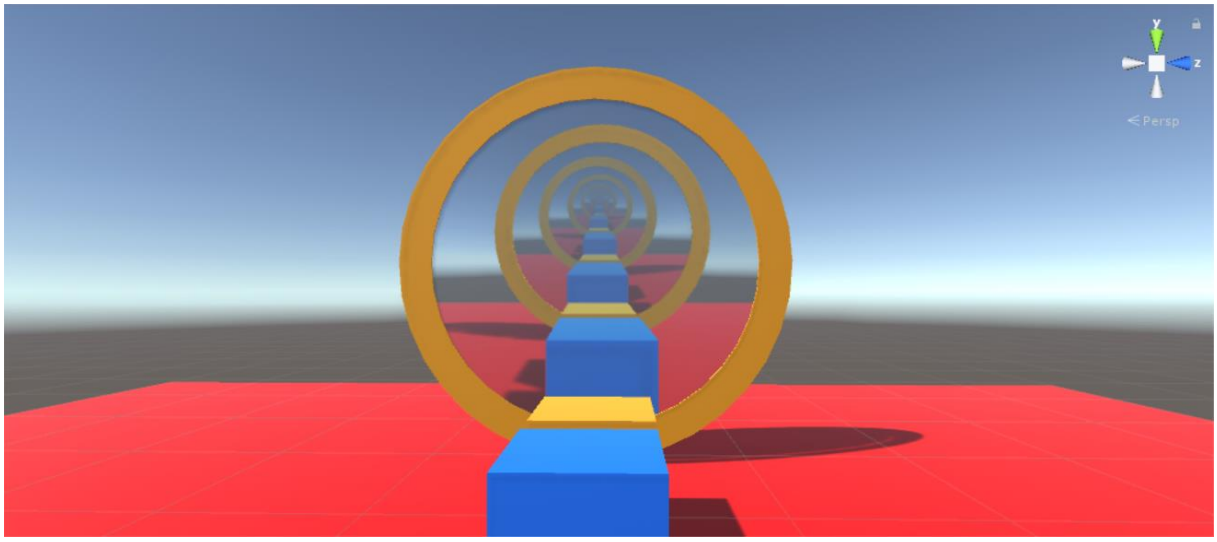
Slika 22. Pikseliziranost teksture



Slika 23. Povećana rezolucija teksture



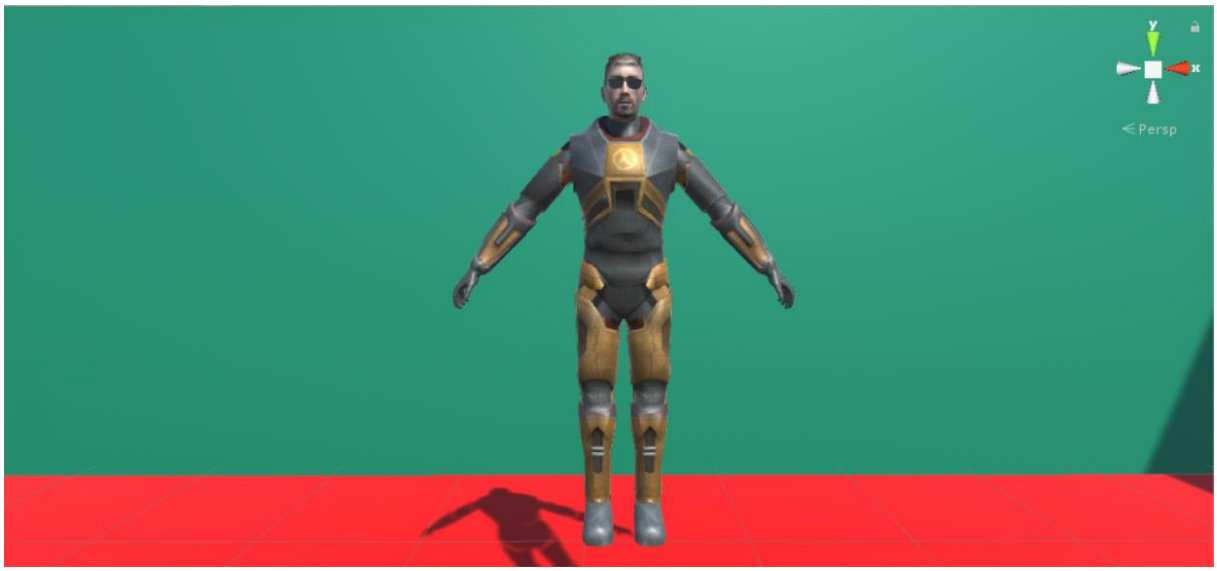
Slika 24. Portali licem u lice u sceni



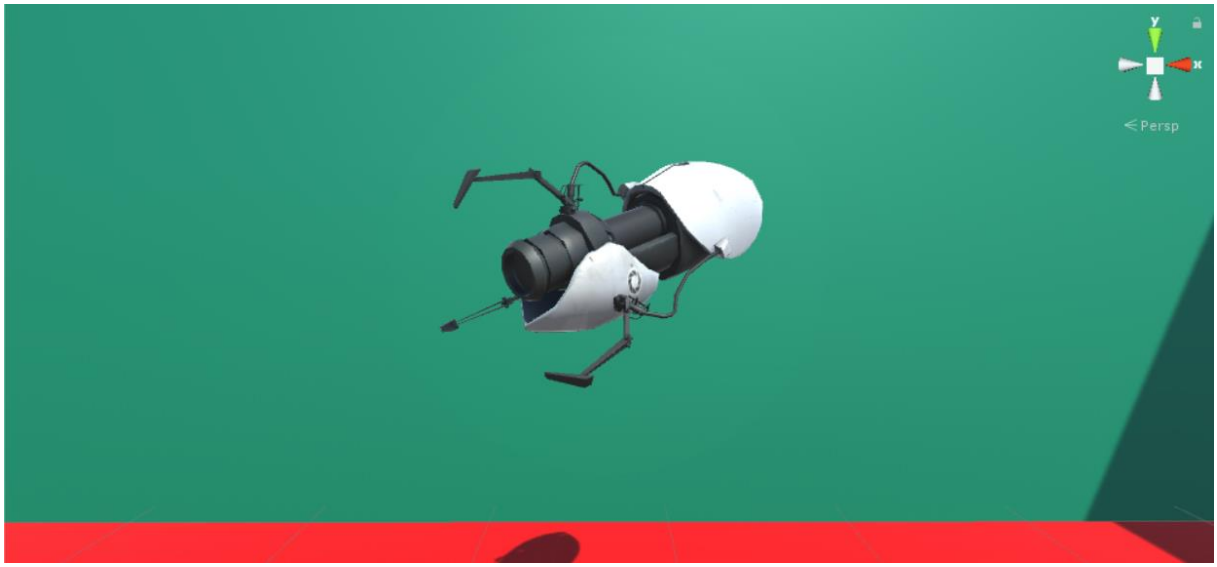
Slika 25. Rekurzivni prikaz portala

4.2 Modeli igrača i oružja za stvaranje portala

Što se tiče modela korištenih za demonstraciju stvaranja portala, korišteni su gotovi modeli igrača (slika 26) i oružja za stvaranje portala (slika 27). [7]

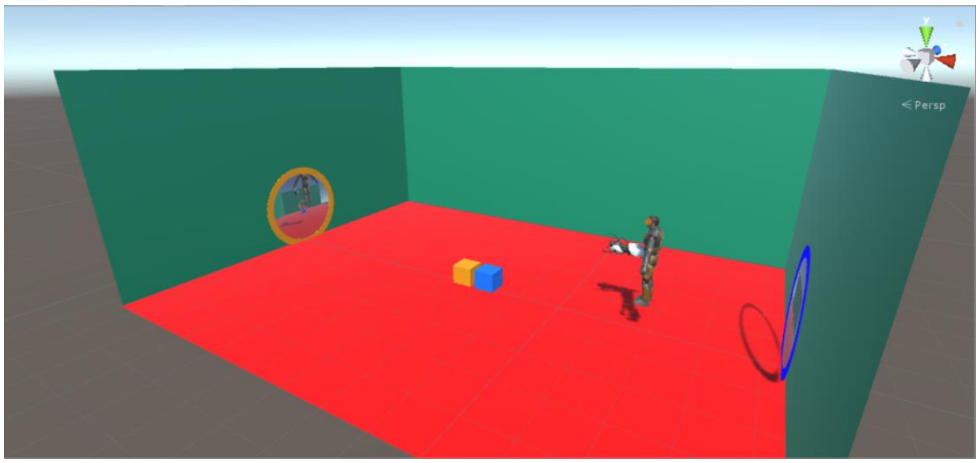


Slika 26. Model igrača



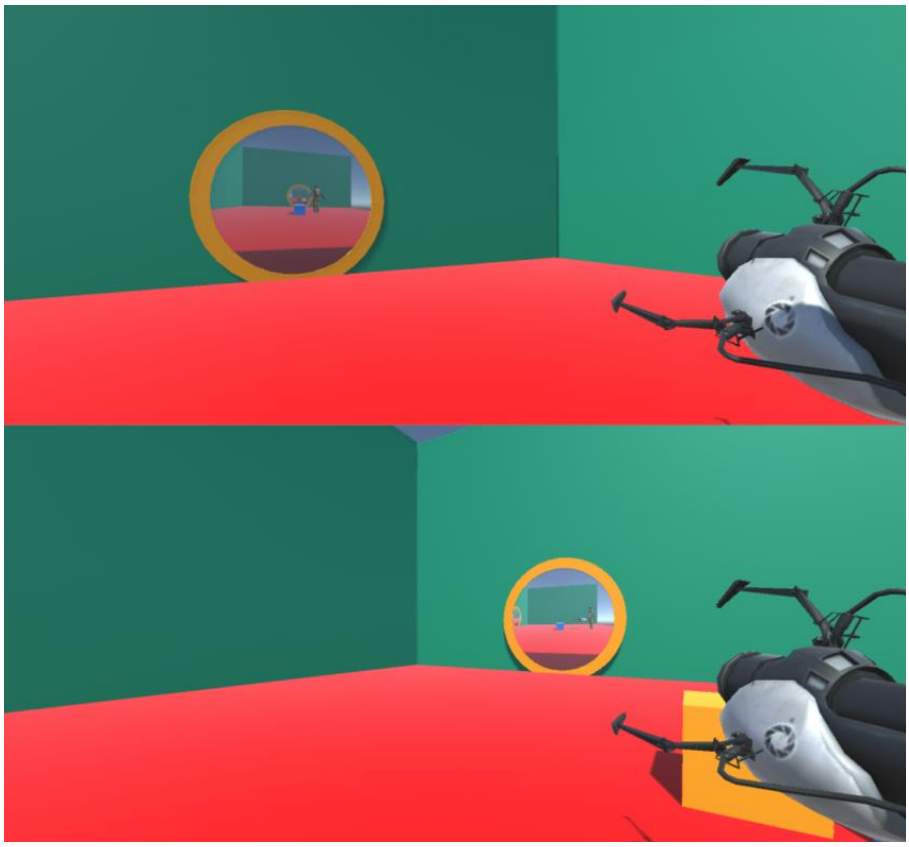
Slika 27. Model oružja za stvaranje portala

4.3 Mišem upravljano stvaranje portala



Slika 28. Prikaz scene s igračem i oružjem za stvaranje portala

Kako bismo mogli prilagođavati mjesto na kojem se nalaze portali, potrebno je napraviti skriptu koja će na pritiske lijevog i desnog gumba miša pomicati portale. Skripta koja će obavljati zadanu funkciju prikazana je u nastavku.



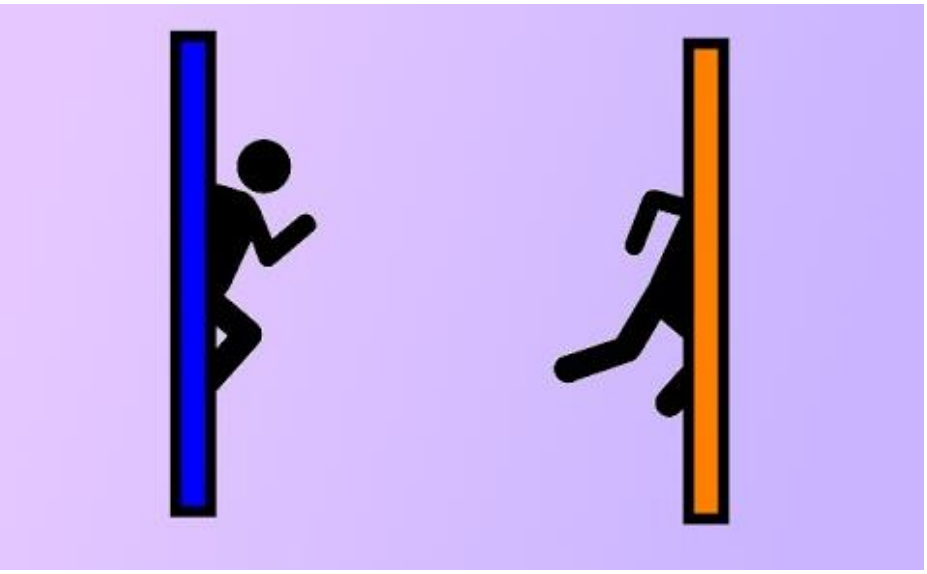
Slika 29. Pritiskom na miš portal pomakne poziciju

```
public class ThrowPortal : MonoBehaviour {

    public GameObject portalA;
    public GameObject portalB;

    GameObject cam;

    // Use this for initialization
    void Start () {
        cam = GameObject.FindWithTag("MainCamera");
    }

    // Update is called once per frame
    void Update () {
        if(Input.GetMouseButtonDown(0))
        {
            ThrowPortals(portalA);
        }
        if (Input.GetMouseButtonDown(1))
        {
            ThrowPortals(portalB);
        }
    }

    void ThrowPortals(GameObject portal)
    {
        int x = Screen.width / 2;
        int y = Screen.height / 2;
        Ray ray = cam.GetComponent<Camera>().ScreenPointToRay(new Vector3(x,
y));
        RaycastHit hit;
        if(Physics.Raycast(ray,out hit))
        {
            Quaternion hitObjectRotation =
Quaternion.LookRotation(hit.normal);
            portal.transform.position = hit.point;
            portal.transform.rotation = hitObjectRotation;
        }
    }
}
```

Isječak programskog kôda 4. Pomicanje portala na lokaciju kursora

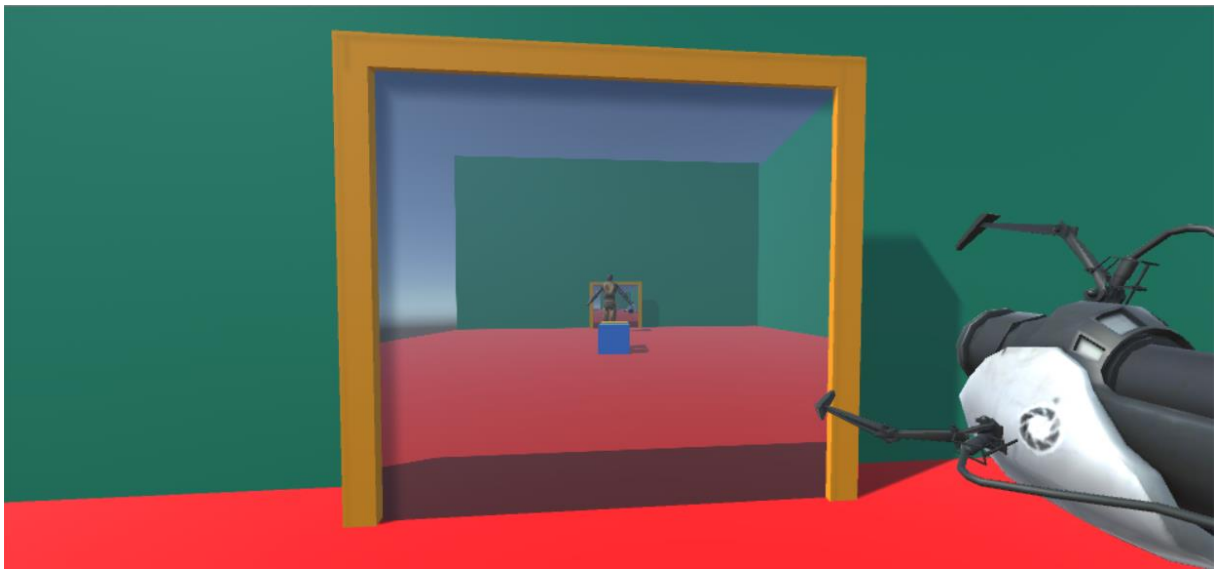
4.4 Fizika portala

Fizika iza portala je zasnovana na mogućnosti teleportacije, tj. prolaskom kroz izvorišni portal igrač izlazi iz odredišnog portala (slika 30). Ta veza između portala može biti jednosmjerna ili dvosmjerna ovisno o vrsti portala.

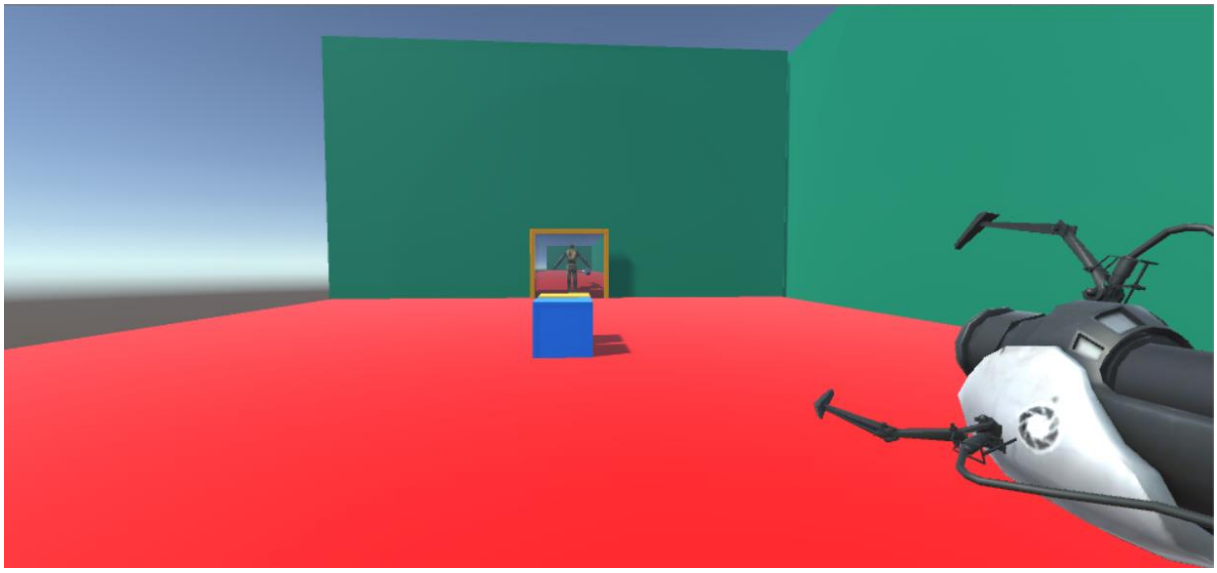


Slika 30. Teleportacija igrača

Na slikama 31. i 32. možemo vidjeti primjer kako to izgleda prije nego što igrač prođe kroz portal i nakon što prođe kroz portal.

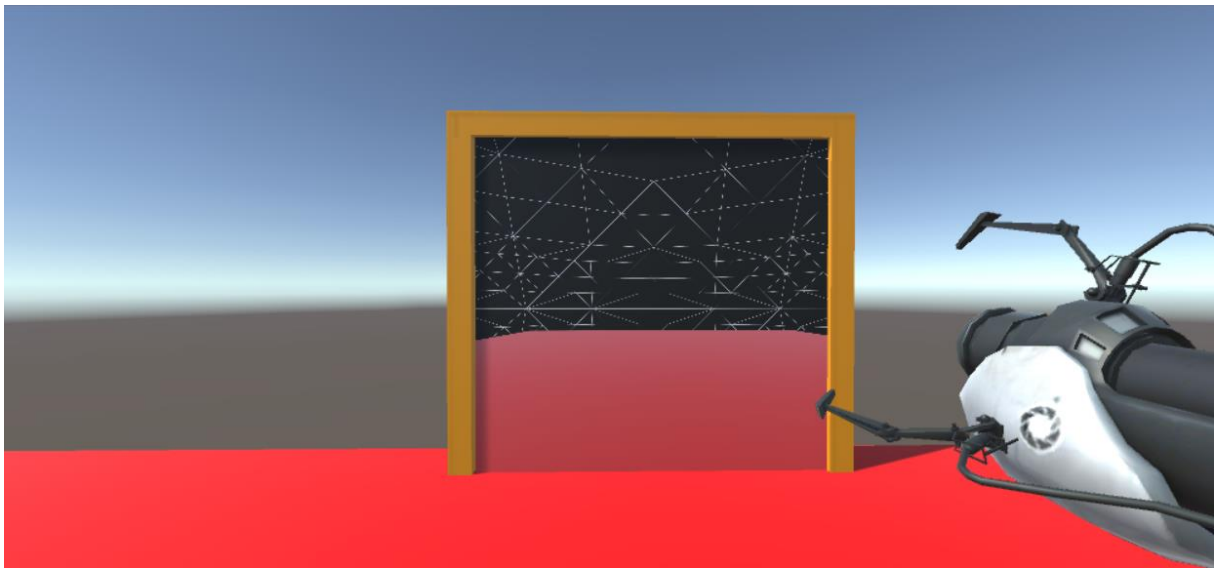


Slika 31. Pozicija igrača prije prolaska kroz portal

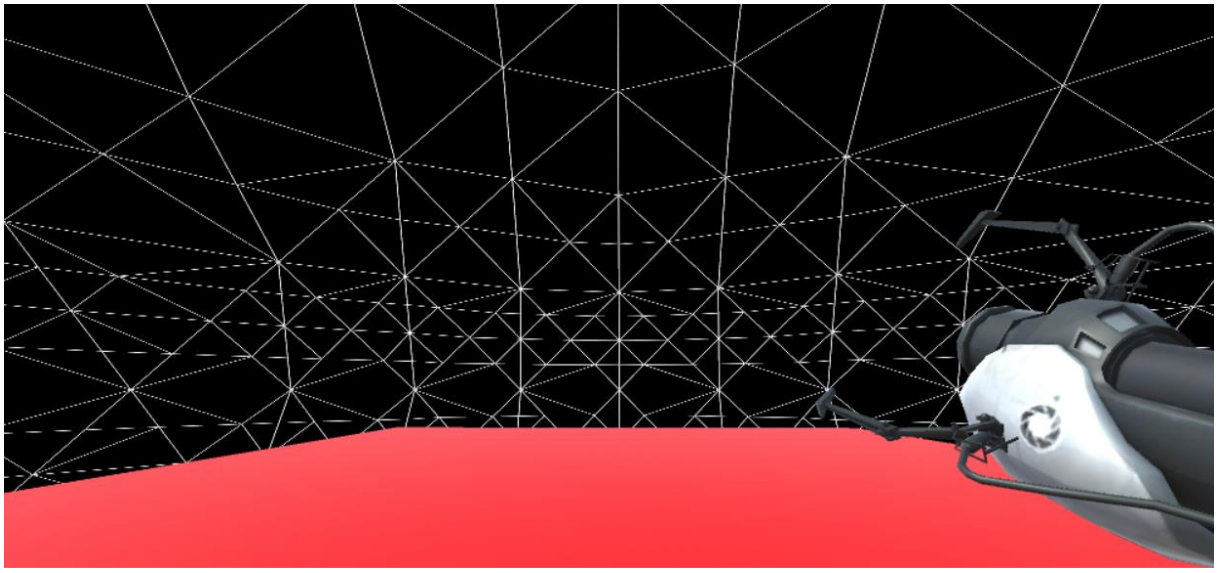


Slika 32. Pozicija igrača nakon prolaska kroz portal

Portali se također mogu koristiti kako bi igrači došli do sljedeće razine (slike 33. i 34.)



Slika 33. Igrač prije prelaska na novu razinu



Slika 34. Igrač nakon prelaska na novu razinu

Mogućnost prolaska kroz portale ostvarena je pomoću transformacije igrača na lokaciju odredišnog portala što možemo vidjeti na isječku programskog kôda 5.

```
public class StepThroughPortal : MonoBehaviour {

    public GameObject otherPortal;

    private void OnTriggerEnter(Collider col)
    {
        if(col.tag == "Player")
        {
            // Transform player position to other portal
            col.transform.position = otherPortal.transform.position -
otherPortal.transform.forward * 5f;
        }
    }

}
```

Isječak programskog kôda 5. Teleportacija igrača na odredišni portal

Popis slika

Popis isječaka programskog kôda

5. Zaključak

U ovome radu obradili smo problematiku iscrtavanja portala. Prvo rješenje iscrtavanja portala objašnjeno je pomoću grafičke biblioteke OpenGL. U tom rješenju su prikazana dva najpopularnija načina iscrtavanja portala, iscrtavanje na teksturu i korištenje spremnika maske. Dok se iscrtavanje na teksturu na prvi pogled čini kao lakši pristup, brzo uviđamo problem kod tog pristupa, a to je rezolucija teksture koja postane pikselizirana ako se igrač previše približi portalu. Metoda spremnika maske omogućava da iscrtamo i unutrašnjost i vanjski dio okvira portala bez korištenja odvojenih ciljeva iscrtavanja. Drugo rješenje napravljeno je pomoću programa Unity u kojem je uz prikaz portala dodano i pomicanje portala pomoću miša te teleportacijska funkcija ako igrač prođe kroz portal. Prednost Unitya nad OpenGL-om je taj što Unity omogućava veću razinu apstrakcije. Zbog toga veći fokus može biti na samom algoritmu za iscrtavanje portala i teleportaciju dok u OpenGL-u fokus mora biti i na samom učitavanju grafičkih objekata, prikazu scene i raznim interakcijama između objekata. Budućnost rada i mogućnost poboljšavanja leži u boljem prikazu unutrašnjosti okvira portala pomoću različitih sjenčara (eng. shaders) te poboljšavanje teleportacijske funkcije tako da prolaz između portala ne bude trenutačan nego da se možemo kretati bez grubih prijelaza koristeći drugačije metode, npr. dupliciranje modela igrača prilikom prolaska kroz portal.

6. Literatura

[1] Matt Blitz, The Grandfather of Video Games You've Never Heard Of, Datum nastanka: 28. ožujka 2016., Datum pristupa: 3. lipnja 2018.

https://www.popularmechanics.com/culture/gaming/a20129/the-very-first-video-game/

[2] Portal rendering | Computer Graphics | FANDOM powered by Wikia, Datum pristupa: 13. lipnja 2018.

http://graphics.wikia.com/wiki/Portal_rendering

[3] OpenGL Programming/Mini-Portal - Wikibooks, open books for an open world, Datum nastanka: 9. kolovoza 2016., Datum pristupa: 4. svibnja 2018.

https://en.wikibooks.org/wiki/OpenGL_Programming/Mini-Portal

[4] Oblique View Frustum Near-Plane Clipping, Datum pristupa: 7. svibnja 2018.

http://www.terathon.com/code/oblique.html

[5] Lengyel, Eric. “Oblique View Frustum Depth Projection and Clipping”. Journal of Game Development, Vol. 1, No. 2 (2005), Charles River Media, pp. 5–16.

[6] Thomas Rinsma, OpenGL Engine Test, Datum nastanka: 15. travnja 2013., Datum pristupa: 4. svibnja 2018

https://github.com/ThomasRinsma/opengl-game-test

[7] 3D Models for Free – Free3D.com, Datum pristupa: 2. lipnja 2018.

https://free3d.com/

[8] OpenGL News Archives, Datum pristupa: 7. svibnja 2018.

https://www.opengl.org/documentation/

[9] Unity - Manual: Unity User Manual (2018.1), Datum pristupa: 25. svibnja 2018.

https://docs.unity3d.com/Manual/index.html

7. Sažetak

Iscrtavanje portala može se raditi na različite načine među kojima su najpopularniji iscrtavanje na teksturu i iscrtavanje korištenjem spremnika maske. U ovom radu prikaz rekurzivnog iscrtavanja portala napravljen je pomoću biblioteke OpenGL te pomoću programa Unity, gdje je dodana i mogućnost teleportacije te kontrole pozicije portala mišem.

Ključne riječi: portal, iscrtavanje, rekurzivno, OpenGL, Unity, teleportacija

Portal rendering can be done with different methods, of which most popular ones are rendering to a texture and using the stencil buffer. In this thesis rendering recursive portals are implemented with OpenGL and game engine Unity, where the possibility of teleportation and controlling portal position with mouse are added.

Keywords: portal, rendering, recursive, OpenGL, Unity, teleportation