SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Računalna animacija

IMPLEMENTACIJA PODRUČJA VIDLJIVOSTI

Josip Komljenović

III. samostalna vježba

U Zagrebu, siječanj 2022.

1. Uvod

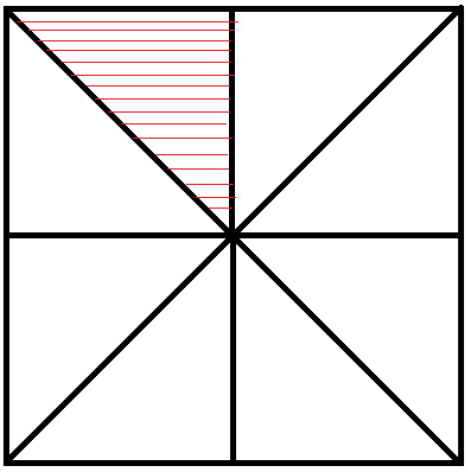
Jedan od problema kojim se bavi područje računalne grafike i animacije je kako vizualizirati područja koja su trenutno vidljiva gledano iz perspektive nekog objekta u prostoru. Ključna prepreka za rješenje tog problema je kako odrediti područja koja se nalaze u sjeni. Kako bi riješili taj problem, osmišljeni su razni algoritmi. U ovom radu će biti obrađena varijanta *shadowcasting* algoritma primijenjena na dvodimenzionalnu mapu promatranu iz ptičje perspektive. Primjene takvog sustava su razne, od vizualizacije rada sustava za autonomno upravljanje, do industrije računalnih igara.

Ova vježba je riješena korištenjem programskog jezika *Python* i biblioteke *Pygame*.

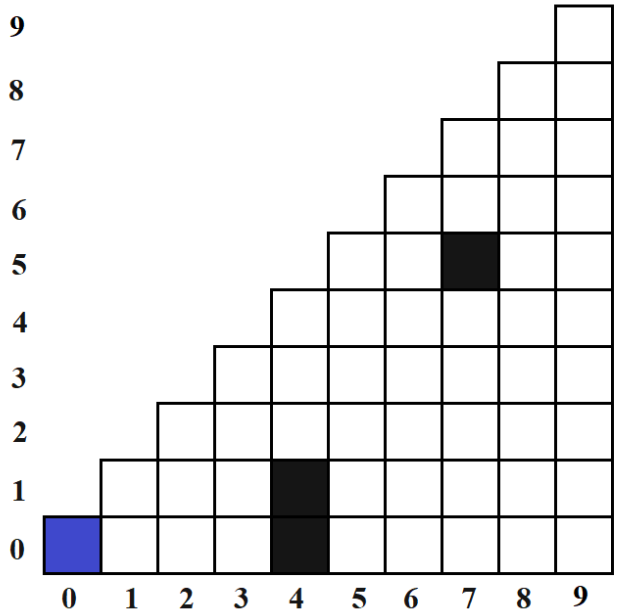
2. Algoritam

Za rješenje prethodno navedenog problema postoje razni algoritmi. Najjednostavniji algoritam je *raycasting* algoritam. Iako *raycasting* algoritam ima primjene u brojnom nizu problema, u ovom slučaju nije najoptimalniji algoritam. Naime, ovisno o broju zraka koje koristimo za određivanje točke sudara s drugim objektima u prostoru, određeni broj točki ćemo provjeriti više puta. To će posebno biti izraženo blizu izvora zraka, a budući da će implementacija koristit relativno malen radijus vidnog polja taj problem će biti poprilično izražen. Stoga je osmišljen *shadowcasting* algoritam. Kod shadowcasting algoritma provjeravaju se horizontalne, odnosno vertikalne linije oko objekta koji ima mogućnost promatranja okolnog prostora i na temelju njih se određuje koji su dijelovi u sjeni. Na taj način su gotovo sve točke provjerene samo jedanput, a tek neznatan broj njih dvaput, što značajno poboljšava performanse programa.

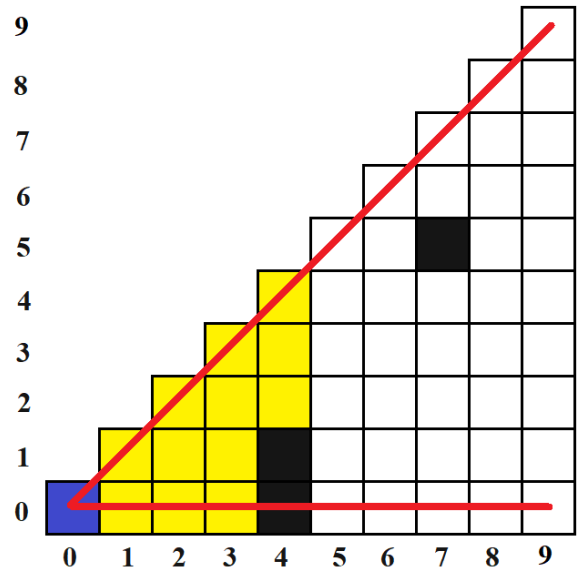
Prvi korak algoritma je podijeliti prostor oko objekta na 8 pravokutnih jednakokračnih trokuta. Zatim se unutar svakog od tih trokuta testiraju horizontalne, odnosno vertikalne linije počevši od središta, kao što je prikazano na sljedećoj slici:



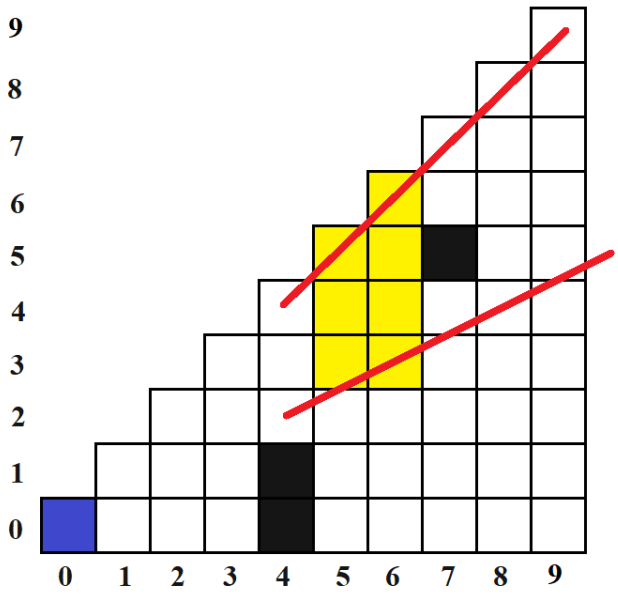
Nakon što smo podijelili prostor na 8 trokuta, nad svakim od njih pozivamo potprogram za generiranje osvijetljenih prostora. Potprogram će biti objašnjen na sljedećem primjeru:



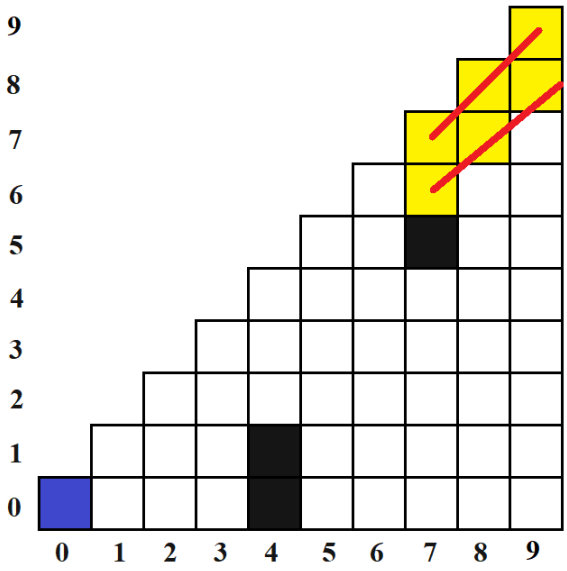
Na početku potprograma postavljamo početne vrijednosti nagiba pravaca koji definiraju osvijetljeni prostor. To su 0 (pravac koji ide od točke (0, 0) do točke (9, 0)) i 1 (pravac koji ide od točke (0, 0) do točke (9, 9)). Time smo definirali interval [0, 1] i sve točke koje su unutar tog intervala (za vrijednost točke računamo $\frac{y}{x}$) obilježavamo kao osvijetljene. U prvom koraku ulazimo u stupac x = 1, zatim x = 2… Postupak nastavljamo do nailaska na prvu prepreku (točka (4, 1)).



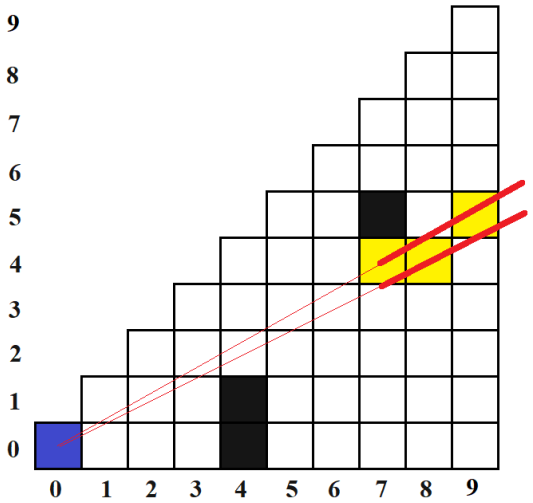
Prilikom nailaska na prepreku izračunavamo nagib pravca u prvoj prethodnoj točki (4, 2) koji iznosi $\frac{2}{4} = 0.5$ i budući da idemo iz osvijetljenog prostora u prepreku mijenjamo početak osvijetljenog intervala koji sada iznosi [0.5, 1]. U sljedećem koraku (za x = 5) posljednja osvijetljena točka je (5, 3), čiji nagib iznosi 0.6, te je taj broj unutar intervala [0.5, 1], međutim već za sljedeća točku (5, 2) je nagib 0.4. Taj nagib se nalazi izvan intervala te se ta točka označava kao skrivena (u sjeni). Postupak se ponavlja do sljedeće prepreke (7, 5).



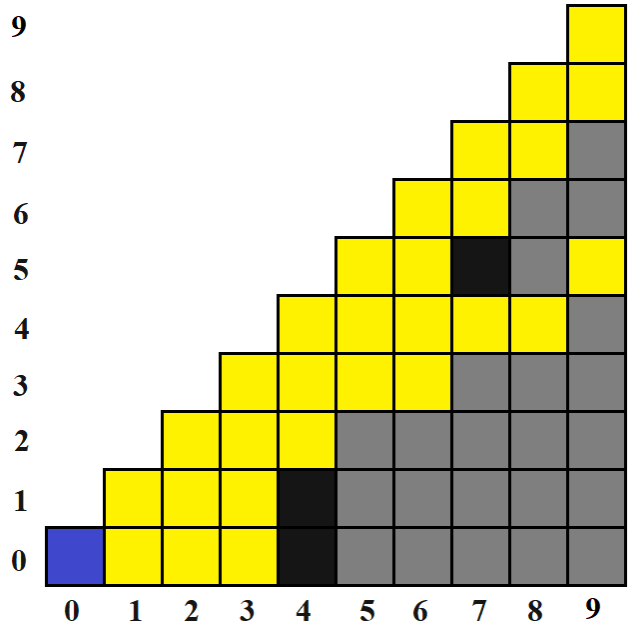
U trenutku nailaska na prepreku interval se ponovno smanjuje (ovoga puta na interval $[\frac{6}{7}, 1]$) te se postupak za taj interval nastavlja dalje do kraja.



Osim toga, prelazak iz prepreke u slobodne točke se nalazi u točki (7, 4). Ta točka je unutar intervala korištenog za x = 7, koji je iznosio [0.5, 1], te se stoga kao osvijetljen interval uzima i dio „ispod“ prepreke (7, 5). Budući da se prelazi iz prepreke u slobodan dio mijenja se krajnja točka intervala, te on sada iznosi [0.5, $\frac{4}{7}$] nakon čega se i postupak za taj interval nastavlja do kraja.



Nakon što je dovršen postupak za x = 9 algoritam je gotov i kao rezultat vraća listu osvijetljenih točaka.

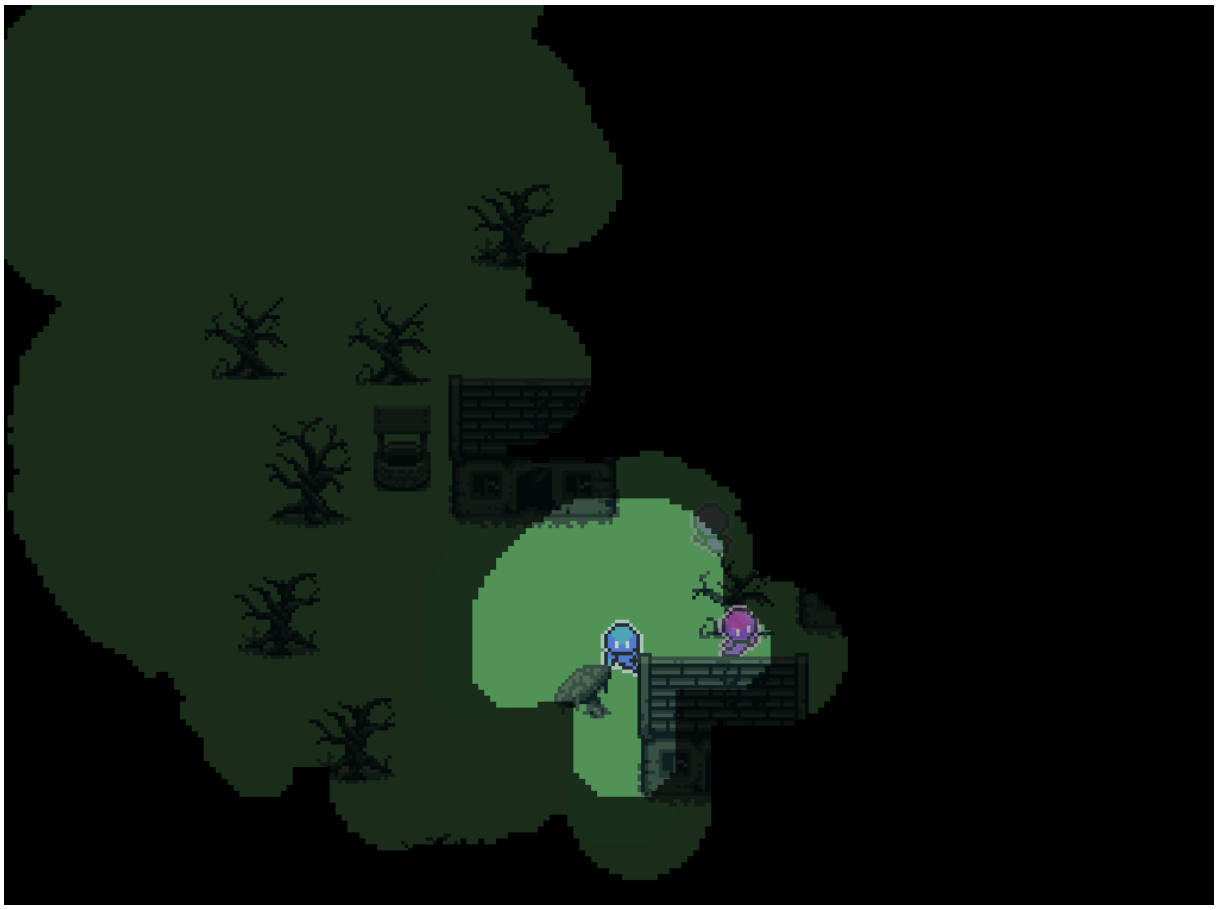


3. Impementacija

Prethodno navedeni algoritam je korišten prilikom otkrivanja prostora koji je osvijetljen. Budući da je željeno vidno polje u projektu u obliku kruga u algoritmu je korištena kvadratna površina kod koje je stranica kvadrata jednaka promjeru kruga, a uz to je i dodana provjera je li dobivena osvijetljena točka unutar radijusa kruga. Točke koje su jednom osvijetljene dodaju se u listu otkrivenih točaka. Na taj način možemo postići i iscrtavanje već otkrivenih područja mape. Prepreke za svijetlost čine razni statični objekti na mapi, poput građevina ili drveća. Osim statičnih objekata, u programu postoje i dinamički objekti. Jedan objekt predstavlja igrača, koji otkriva mapu i koji predstavlja izvor svijetlosti. Također tu je i niz drugih objekata koji se pomiču između nasumično odabranih točki. Ti objekti se iscrtavaju samo ako su u blizini igrača, a inače su skriveni. Prilikom iscrtavanja prvo se iscrtavaju već otkriveni dijelovi mape, a zatim se preko njih iscrtava površina koja skriva objekte. Konačnu boju dobivamo kao umnožak dvije boje podijeljen s 255. Tako za skrivene dijelove kao drugu boju koristimo (0, 0, 0). Time nakon množenja ostvarujemo da se iscrtava crna boja (odnosno 0, 0, 0). Već otkrivena područja koja su dalje od igrača želimo da se iscrtaju tamnije nego što su to originalno. Za to koristimo boju (64, 64, 64), pa se tako boja koja je originalno iznosila (100, 180, 100) iscrtava kao (25, 45, 25). Za područja blizu igrača originalnu boju kombiniramo s (255, 255, 255).

Programski kod je dostupan na sljedećem linku

Primjer rada programa je vidljiv na sljedećoj slici



4. Literatura

- http://www.roguebasin.com/index.php/FOV_using_recursive_shadowcasting