SVEUCILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RACUNARSTVA

SAMOSTALNA LABORATORIJSKA VJEŽBA

Računalna animacija

Simulacija Sunčevog sustava

Andrija Župić

Zagreb, siječanj, 2022.

Sadržaj

1. Opis projekta

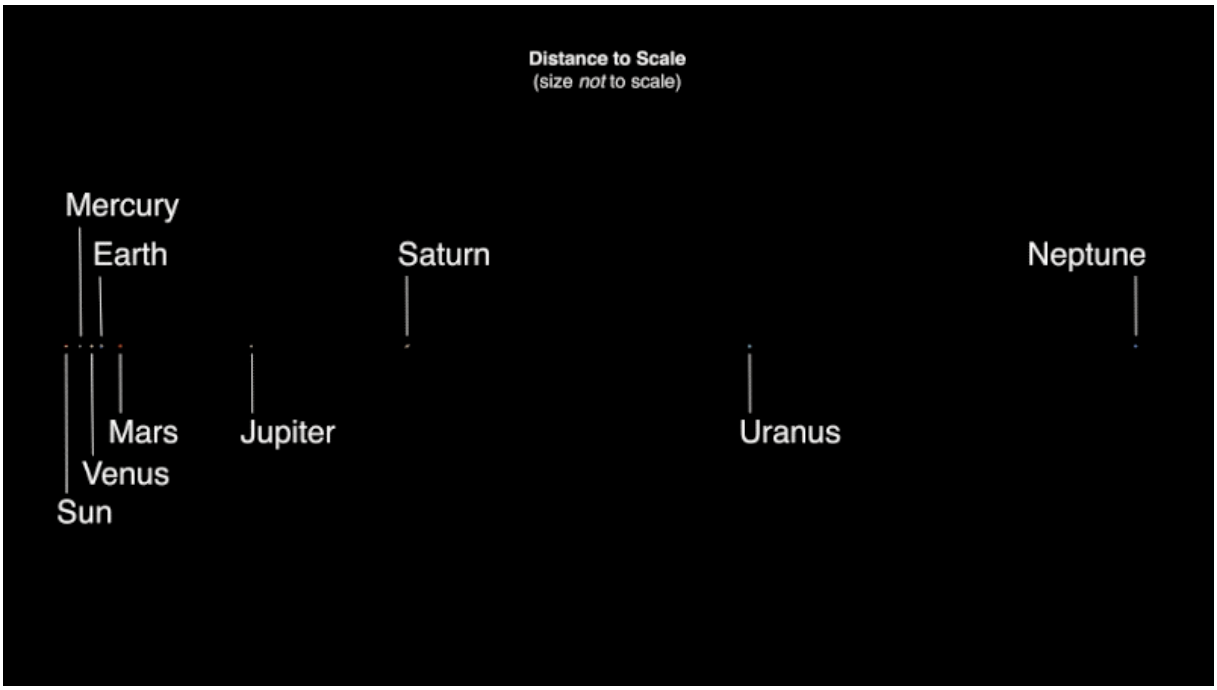
Tema ovog projekta je simulacija Sunčevog sustava[1] u Python programskom jeziku. Sunčev sustav sastoji se od Sunca i 8 planeta: Merkur, Venera, Zemlja, Mars, Jupiter, Saturn, Uran i Neptun.

Svaki planet ima svoju vlastitu brzinu rotacije (okretanje oko svoje osi) i revolucije (okretanje oko nekog drugog objekta, u ovom slučaju Sunca). Relativni odnosi brzine rotacije i revolucije planeta su očuvani s obzirom na stvarne vrijednosti, uz očitu napomenu da su stvarne vrijednosti brzine puno sporije u odnosu na one u programskom rješenju. Tako primjerice Zemlji treba 1 godina da napravi jednu revoluciju oko Sunca, dok joj u programu za to treba tek par minuta.

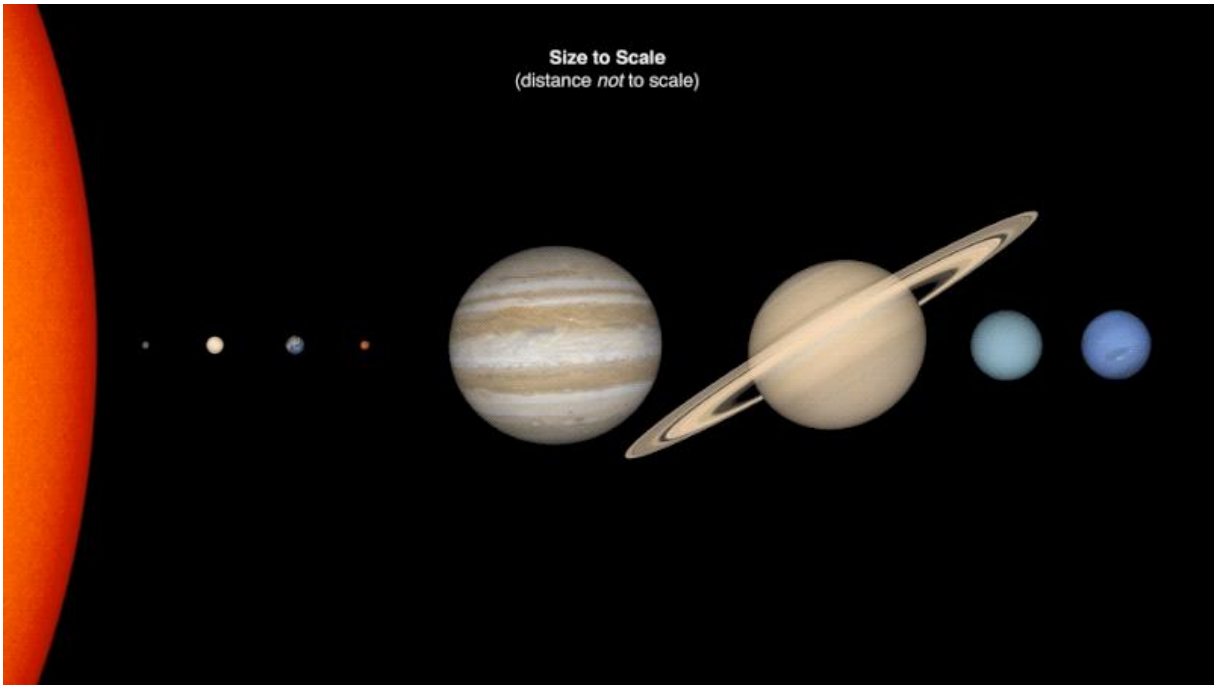
Relativni odnosi udaljenosti planeta od Sunca (*Slika 2*) su također očuvani u odnosu na stvarne udaljenosti (stupac `orbit_scale` u datoteci `planets.csv`). Jedina veličina koja nije očuvana je relativni odnos veličina planeta (stupac `size_scale` u datoteci `planets.csv`) (*Slika 3*). Razlog tomu je taj što bi u slučaju ispravne reprezentacije manji planeti bili jako teško uočljivi. Primjerice, promjer Merkura je 4,879 km, dok je promjer Jupitera 142,984 km. To znači da bi u Jupiter planet Merkur stao 24,462 puta.

U programskom rješenju korišteni su paketi `PyOpenGL`[2] i `Pygame`[3]. Korisniku je omogućeno kretanje u sceni pomoću tipkovnice i miša.

Slika 1. Snimka zaslona razvijenog programskog rješenja simulacije Sunčevog sustava.



2. Relativni odnosi udaljenosti planeta od Sunca[4]



Slika 3. Relativni odnos veličina planeta[4]

2. Implementacija

2.1. Objekti

Jedini objekti korišteni u programskom rješenju su sfere i diskovi. Sfere su korištene za reprezentaciju planeta i pozadinskog neba, dok su diskovi korišteni za iscrtavanje orbita planeta i reprezentaciju Saturnovog prstena. Objekti su skalirani pomoću funkcije `glScale`[17]. Orbite planeta tj. diskovi su skalirane pomoću stupca `orbit_scale`, a sami planeti tj. sfere pomoću stupca `size_scale` u dataframeu `planets`

Za iscrtavanje sfere korištena je `OpenGL` funkcija `gluSphere`[5]. `gluSphere` crta sferu zadanog radijusa (parametar `radius`) sa središtem oko ishodišta. Sfera je podijeljena oko osi z na kriške (parametar `slices`), a duž z osi na hrpe (parametar `stacks`).

Za iscrtavanje diska korištena je `OpenGL` funkcija `gluDisk`[6]. `gluDisk` prikazuje disk na ravnini $z = 0$. Disk ima vanjski radijus (parametar `outer`) i sadrži koncentričnu kružnu rupu definiranu unutarnjim radijusom (parametar `inner`). Ako je `inner` $= 0$, tada rupa u disku ne postoji. Disk je podijeljen oko osi z na kriške (parametar `slices`) i na kolutove (parametar `loops`). Za obje funkcije je također potreban i kvadrični objekt koji se stvara pomoću funkcije `gluNewQuadric`.

2.2. Teksture

Teksture za Sunce, planete i pozadinsko nebo preuzete su sa *Solar System Scope*[7] stranice. Teksture u ovom paketu temelje se na NASA-inim podacima. Boje i nijanse tekstura su podešene prema fotografijama u pravim bojama koje su napravile svemirske letjelice *Messenger*, *Viking*, *Cassini* i svemirski teleskop *Hubble*.

Imena tekstura su u programu spremljena u polje pomoću funkcije `glGenTextures`[8]. Veličina polja se inicijalizira na broj teksturalnih datoteka. Nakon toga pomoću funkcije `glBindTexture`[9] vežemo ime teksture na ciljnu teksturu. Na posljetku pomoću funkcije `gluBuild2DMipmaps`[10] gradimo dvodimenzionalnu mipmapu. Teksturalne datoteke učitavamo pomoću `image`[11] modula `Pygame` paketa. Kako bi primijenili teksture na stvorene objekte koristimo funkciju `gluQuadricTexture`[12] kojoj predajemo objekt sfere ili diska i zastavicu `True` kojom govorimo da želimo da se generiraju teksturalne koordinate.

2.3. Sjenčanje

Sjenčanje je ostvareno pomoću `OpenGL` funkcije `glLightfv`[13] kojoj prvo predajemo izvor svjetlosti `GL_LIGHT0` i postavljamo ga na `GL_DIFFUSE` te specificiramo RGBA intenzitet svjetla. Nakon toga `GL_LIGHT0` postavljamo u ishodište (Sunce) opet pomoću `glLightfv` funkcije, parametra `GL_POSITION` i koordinata pozicije svjetla.

Kako bismo omogućili sjenčanje moramo još omogućiti sustav osvjetljenja pomoću `glEnable(GL_LIGHTING)` i `glEnable(GL_LIGHT0)`. U programu je postavljeno da osvjetljenje ne utječe na Sunce i pozadinsko nebo jer se u suprotnom slučaju teksture tih objekata potamne.

2.4. Vremenska evolucija

Rotacija i revolucija planeta definirane se pomoću rotacijskog i revolucijskog koraka u `planets.csv` datoteci. Veličine u datoteci su takve da ako imamo 30 ažuriranja položaja u sekundi, tada jedna revolucija Zemlje oko Sunca tj. jedna godina u programu traje 1 minutu u stvarnom vremenu. Međutim, u programu je ta brzina usporena jer su plinoviti planeti (Jupiter, Saturn, Uran, Neptun) imali toliko veliku brzinu rotacije da je došlo do pojave vremenskog aliasa. Zbog toga je sada jedna revolucija Zemlje oko Sunca u programu usporena na 10 minuta u stvarnom vremenu. Inicijalni podaci iz datoteke `planets.csv` se u programu učitavaju u `pandas dataframe` i tamo se tijekom izvođenja programa ažuriraju.

Pri svakom osvježavanju scene (30 puta u sekundi) se iz dataframea `planets` učitavaju `revolution_step` i `rotation_step` koji predstavljaju korake revolucije i rotacije te `curr_rev` i `curr_rot` u kojima se nalaze trenutna pozicija planeta na orbiti i trenutni kut rotacije planeta oko svoje osi. Vrijednosti za trenutnu poziciju revolucije i kut rotacije planeta ažuriraju se pri svakom osvježavanju scene.

Pozicija planeta na orbiti se ažurira za osi x i z pomoću funkcije sinusa, odnosno kosinusa i trenutne pozicije na orbiti `curr_rev`. Pomoću dobivenih koordinata translacije `translateX` i `translateZ` i funkcije `glTranslate`[14] na trenutnu matricu primjenjuje se (množenjem) translacijska matrica: $1\ 0\ 0\ x\ 0\ 1\ 0\ y\ 0\ 0\ 1\ z\ 0\ 0\ 0\ 1$.

Kut rotacije planeta se ažurira pomoću funkcije `glRotatef`[15] tako da se na trenutnu matricu primjeni matrica rotacije za kut `curr_rot` oko vektora `rot_axX`, `rot_axY`, `rot_axZ` koji je također učitan iz dataframea `planets`. Vidimo da se svi planeti rotiraju oko osi y osim

Urana koji je rotira bočno oko osi x. Pri iscrtavanju planeta sfere također treba rotirati oko osi x za 90^o (u slučaju Urana oko osi y) kako bi bili ispravno prikazani u sceni (definirano u stupcima `tiltX`, `tiltY` i `tiltZ` u dataframeu `planets`).

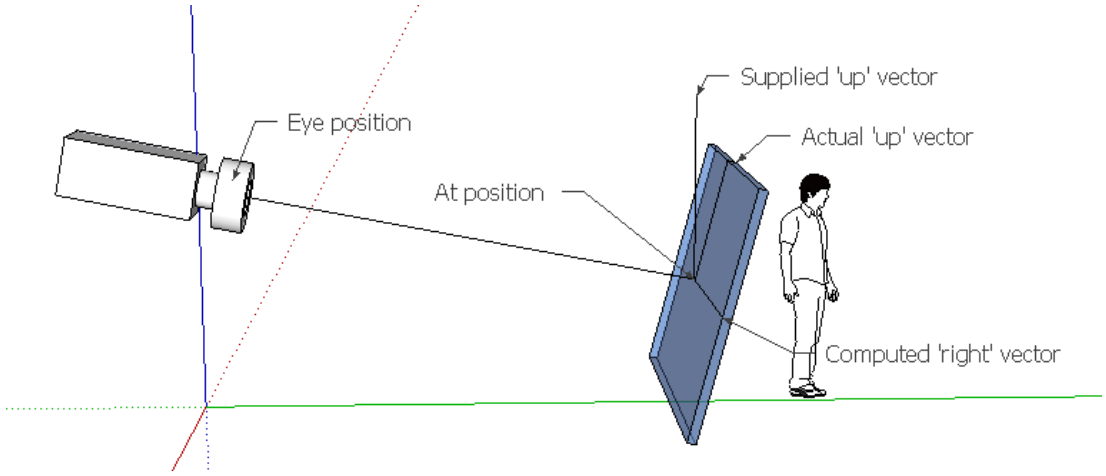
2.5. Kretanje u sceni

Kretanje u sceni omogućeno je pomoću tipkovnice i miša. U funkciji `keys` definirano je ponašanje za tipke `W`, `A`, `S`, `D` i pokrete miša. Pritiskom na tipku `W` kameru pomičemo u sceni prema naprijed, tipkom `A` prema lijevo, tipkom `S` prema natrag i tipkom `D` prema desno.

S obzirom da gledište kamere također ovisi i mišu, pomicanje kamere u sceni „prema naprijed“, „prema natrag“ itd. relativni su pojmovi koji ovise o trenutnom položaju miša. Zato se iz dataframea `settings` učitava i trenutni vektor položaja miša `mouseAngleX`, `mouseAngleY`. Pomoću tog vektora se određuje novi položaj kamere u sceni opisan vektorom `eyeX`, `eyeY`, `eyeZ` koji se opet ažurira u `settings` dataframeu.

Pomicanjem miša mijenja se referentna točka koja predstavlja centar scene. Ona je određena vektorom `centerX`, `centerY`, `centerZ` koji se određuje pomoću trenutnog položaja kamere u sceni (`eye` vektor) i trenutnog položaja miša (vektor `mouseAngle`). Novoizračunati vektor centra scene se također onda ažurira u dataframeu `settings`.

Sada kada znamo položaj kamere i referentne točke centra u sceni možemo izvesti transformaciju pogleda scene pomoću funkcije `gluLookAt`[16]. Ona osim `eye` i `center` vektora također prima i `UP` vektor kojim je određen smjer „gore“ u sceni (u našem slučaju je „gore“ u smjeru pozitivne osi y: 0, 1, 0).



Slika 4. Transformacija pogleda scene pomoću 3 vektora: eye vektor (“Eye position”), center vektor ("At position") i UP vektor (“Supplied “up” vector”)[18]

3. Upute za pokretanje

Program se pokreće iz terminala naredbom `python solar_system.py` ili običnim pokretanjem iz razvojne okoline. Paketi potrebni za rad programa su `PyOpenGL`, `Pygame` i `pandas`. Početne postavke definirane su u datoteci `settings.csv`: `width` i `height` određuju veličinu prozora za prikaz, a nakon toga su određeni inicijalne vrijednosti vektora `eye`, `center` i `mouseAngle`. Početne vrijednosti vektora su proizvoljne, a ove trenutne su odabrane tako da se pri samom pokretanju programa na ekranu vide Sunce i svi planeti.

Kretanje u sceni moguće je mišem i tipkovnicom. Pritiskom na tipku `W` kameru pomičemo u sceni prema naprijed, tipkom `A` prema lijevo, tipkom `S` prema natrag i tipkom `D` prema desno. Pomicanjem miša prema gore pomičemo liniju pogleda prema gore, pomicanjem miša prema lijevo pomičemo liniju pogleda prema lijevo itd.

Sažetak

Razvijeno programsko rješenje predstavlja simulaciju Sunčevog sustava u Python programskom jeziku. Sunčev sustav sastoji se od Sunca i 8 planeta: Merkur, Venera, Zemlja, Mars, Jupiter, Saturn, Uran i Neptun. Sačuvani su relativni odnosi brzina rotacija, revolucija i udaljenosti planeta od Sunca u odnosu na stvarne vrijednosti. Korisniku je omogućeno kretanje u sceni pomoću tipkovnice i miša.

Summary

The developed software solution is a simulation of the solar system in the Python programming language. The solar system consists of the Sun and 8 planets: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus and Neptune. The relative relations of rotation and revolution speeds, as well as distances of the planets from the Sun in relation to the actual values have been preserved. The user is allowed to move through the scene using a mouse and a keyboard.

Literatura

1. Solar System – Wikipedia: https://en.wikipedia.org/wiki/Solar_System
2. PyOpenGL -- The Python OpenGL Binding: http://pyopengl.sourceforge.net/
3. Pygame Front Page — pygame v2.1.1 documentation: https://www.pygame.org/docs/
4. Student Video: Solar System Size and Distance | NASA/JPL Edu: https://www.jpl.nasa.gov/edu/learn/video/solar-system-size-and-distance
5. gluSphere: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluSphere.xml
6. gluDisk: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluDisk.xml
7. Solar Textures | Solar System Scope: https://www.solarsystemscope.com/textures/
8. glGenTextures: https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glGenTextures.xhtml
9. glBindTexture: https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glBindTexture.xhtml
10. gluBuild2DMipmaps: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluBuild2DMipmaps.xml
11. pygame.image — pygame v2.1.1 documentation: https://www.pygame.org/docs/ref/image.html
12. gluQuadricTexture: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluQuadricTexture.xml
13. glLight: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glLight.xml
14. glTranslate: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glTranslate.xml
15. glRotatef: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glRotate.xml
16. gluLookAt: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluLookAt.xml
17. glScale: https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glScale.xml
18. gluLookAt explanation - Stack Overflow: https://stackoverflow.com/questions/5717654/glulookat-explanation