Deformacija 2D objekta
Tehnička dokumentacija
Verzija 1.0

Josip Sito

Nastavnik: Prof.dr.sc. Željka Mihajlović

Sadržaj

Tehnička dokumentacija

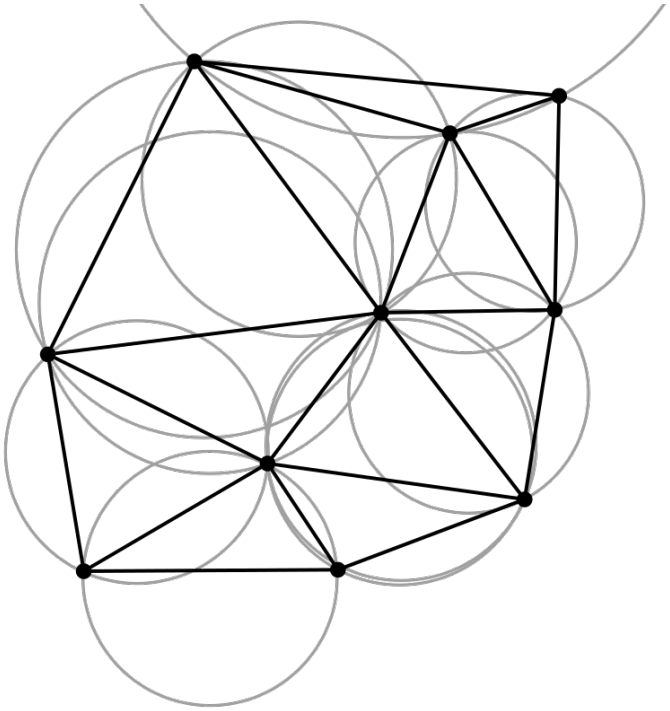
1. Opis razvijenog proizvoda

Cilj projekta je bio istražiti i implementirati deformaciju 2D objekta tako da korisnik sustava mijenjanjem vrijednosti različitih parametara mijenja stanje deformabilnog objekta. Projekt je izrađen u grafičkom alatu Unity, a izvorni kod je napisan u C# programskom jeziku. Deformacija objekta sastoji se od dvije faze: priprema geometrije i animiranje deformacije. Priprema geometrije se odvija pomoću Delaunayeve triangulacije, a animiranje deformacije se temelji na osnovi sustava masa i opruga i semi-implicitne Eulerove metode. Za potrebe testiranja i demonstracije sustava dodana je mogućnost jednostavne detekcije kolizije.

2. Tehničke značajke

2.1 Delaunayeva triangulacija

Za pripremu geometrije korištena je Delaunayeva triangulacija koja predstavlja jedinstvenu podjelu planarnog objekta na trokuta tako da se nijedna točka ne nalazi u opisanoj kružnici bilo kojeg trokuta u triangulaciji (Slika 1).



Slika 1. Primjer Delaunayeve triangulacije

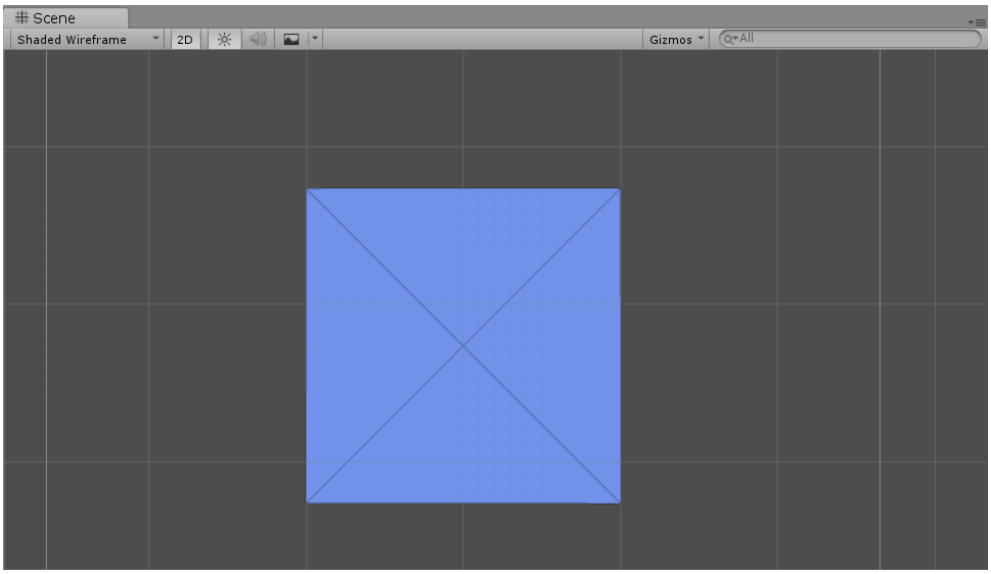
Kao implementacija Delaunay-ove triangulacije korišten je inkrementalan Bowyer-Watson algoritam koji funkcionira tako da dodaje jednu po jednu točku u triangulaciju te je pseudokod algoritma dan u nastavku.

```
 function BowyerWatson (pointList)
      // pointList is a set of coordinates defining the points to be triangulated
      triangulation := empty triangle mesh data structure
      add super-triangle to triangulation // must be large enough to completely
contain all the points in pointList
      for each point in pointList do // add all the points one at a time to the
triangulation
         badTriangles := empty set
         for each triangle in triangulation do // first find all the triangles that
are no longer valid due to the insertion
```

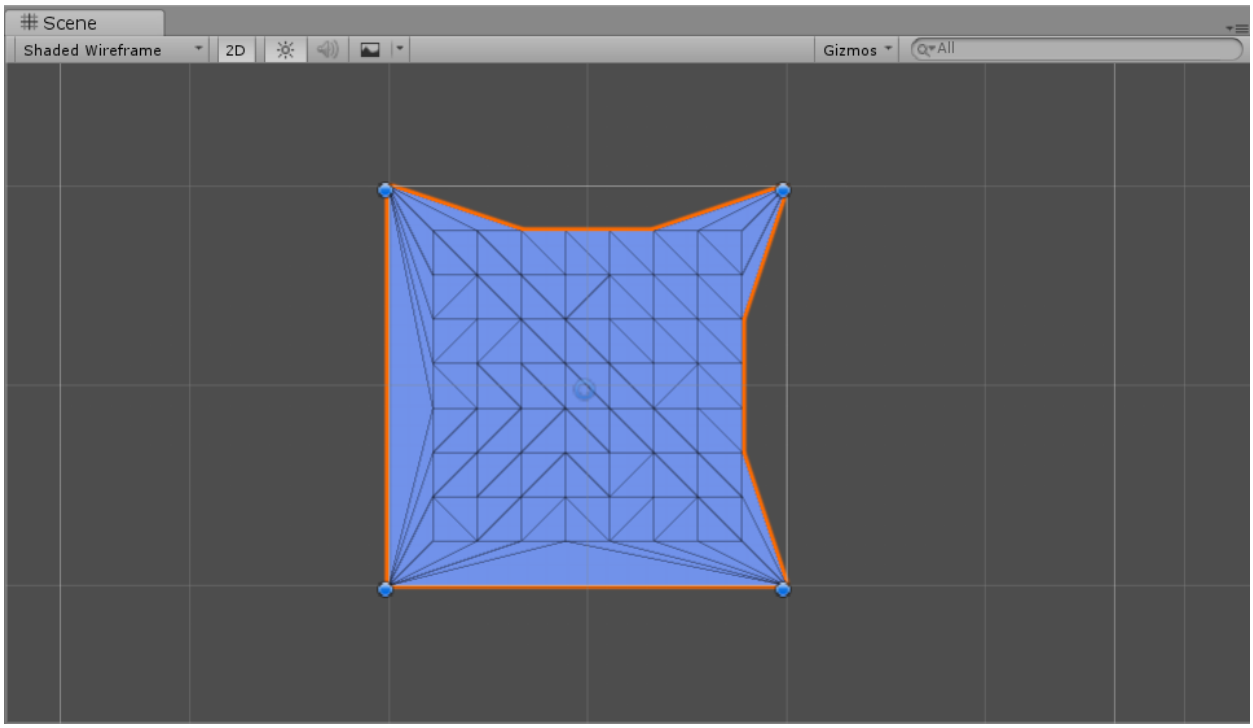
```
            if point is inside circumcircle of triangle
               add triangle to badTriangles
         polygon := empty set
         for each triangle in badTriangles do // find the boundary of the polygonal
hole
            for each edge in triangle do
               if edge is not shared by any other triangles in badTriangles
                  add edge to polygon
         for each triangle in badTriangles do // remove them from the data structure
            remove triangle from triangulation
         for each edge in polygon do // re-triangulate the polygonal hole
            newTri := form a triangle from edge to point
            add newTri to triangulation
      for each triangle in triangulation // done inserting points, now clean up
         if triangle contains a vertex from original super-triangle
            remove triangle from triangulation
      return triangulation
```

Izvor: https://en.wikipedia.org/wiki/Bowyer%E2%80%93Watson_algorithm

Sustav nudi mogućnost stvaranja objekta u obliku kvadra ili objekta pomoću teksture. Kvadar se stvara na način da se zadaju četiri inicijalne točke koje ga definiraju, a zatim se deterministički dodaju točke unutar njega ovisno o odabranoj razini detalja, odnosno za *n* razinu detalja uniformno će se dodati n^2 točaka unutar kvadra (slika 2 i slika 3). Na slici 3 zbog načina na koji su točke početne točke postavljene objekt koji je stvoren neće imati pravilni izgled kvadra.

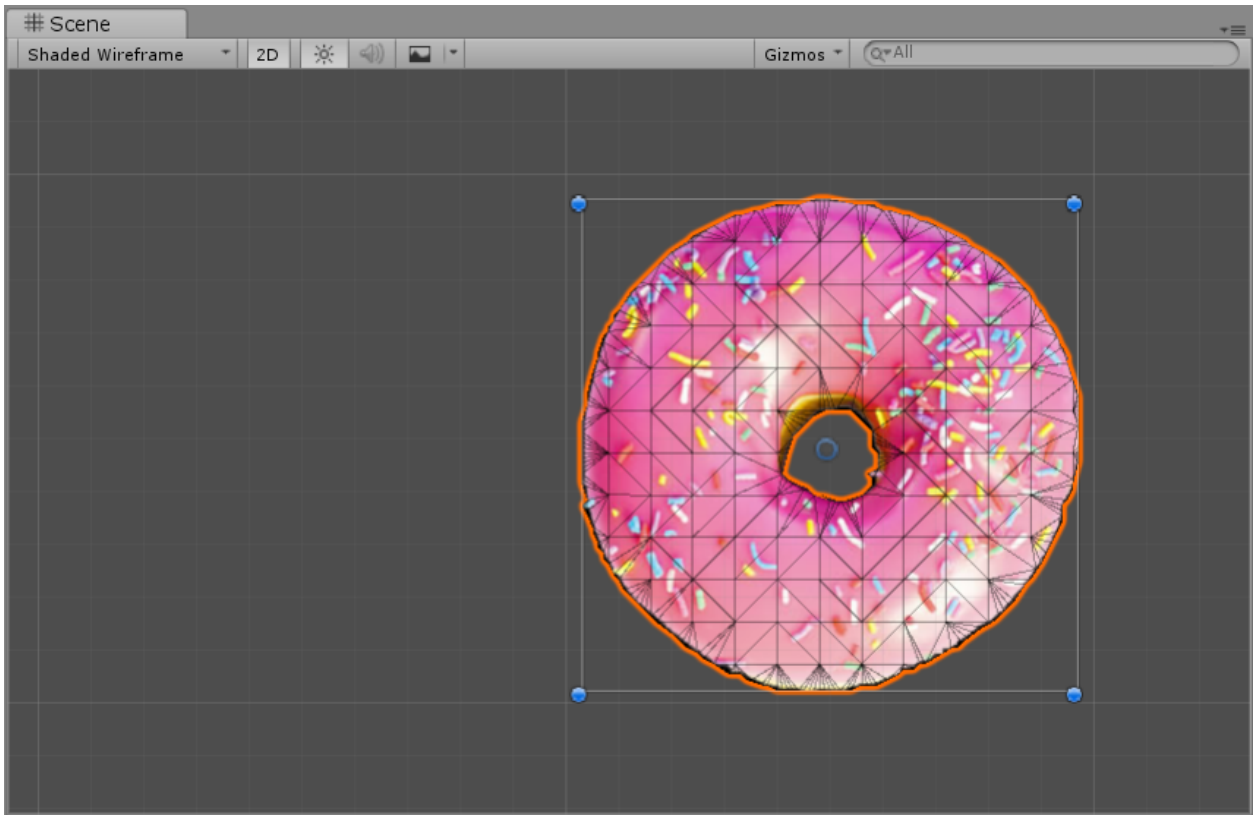


Slika 2. Kvadar sa razinom detalja 1



Slika 3. Kvadar sa razinom detalja 8

Objekt kreiran pomoću teksture stvara se na način da se po slici prati piksel po piksel te ukoliko piksel nije crne boje, onda se doda nova točka na tom položaju. Kako se nebi stvorilo previše točaka zadaje se određeni cijeli broj *n* na temelju kojeg se preskače svakih *n* piksela. Od tekstura može se odabrati slika zeca, krafne ili armadila (slika 4).



Slika 4. Objekt stvoren pomoću teksture krafne

2.2 Sustav masa i opruga

Sustav masa i opruga jedan je od jednostavnijih fizičkih deformacijskih modela. Temeljen je na Hookeovom zakonu $F = -k\Delta x$, a u sustavu je svakoj točki dodana masa. Sila opruge u točki *i* računa se za svaku točku *j* s kojom je točka *i* povezana na temelju sljedećih formula:

$$f_{ij} = k_s(|x_{ij}| - l_{ij})\frac{x_{ij}}{|x_{ij}|} + k_d \cdot v_{ij}$$

k_s- konstanta opruge
x_{ij}- vektor između točke *i* i *j* $(x_j - x_i)$
l_{ij}- početna duljina između točke *i* i *j*
k_d- konstanta prigušenja
v_{ij}-razlika projeciranih brzina *i* i *j* točke na vektor duljine opruge $(v_j - v_i)$

Ukupna unutarnja sila u točki *i* računa se prema:

$$F_i^{internal} = \sum_{i=j_1}^{j_q} f_{ij}$$

Ukupna sila koja djeluje na točku *i*:

$$F_i = F_i^{internal} + F_i^{external}$$

Gdje vanjska sila može biti:

$$F_i^{external} = m \cdot g$$

2.3 Detekcija sudara

Nakon računanja sile provodi se detektiranje kolizije. Implementiran je jednostavan sustav detekcije kolizije gdje se za svaku točku računa sudara li se ona s objektom u obliku pravokutnika. Za detekciju kolizije pravokutnika s točkom korišten je sljedeći pseudokod:

```
function IsColliding (P)
        // points of rectangle are A, B, C, D
        AB := vector edge from A to B
        BC := vector edge from B to C
        AP := vector from A to P
        BP := vector from B to P
        return 0 <= dot(AB, AP) && dot(AB, AP) <= dot(AB, AB) && 0 <= dot(BC, BP) &&
        dot(BC, BP) <= dot(BC, BC)
```

Nakon što se uspostavi da je došlo do sudara točka se vrati na položaj prije sudara, dohvati se normala ruba pravokutnika kod kojeg je došlo do sudara te se brzina u smjeru suprotnom od dohvaćene normale postavi na nulu. Također sve dok je točka u koliziji sa pravokutnikom na nju djeluje sila trenja u smjeru suprotnom od smjera gibanja te se brzina točke smanjuje ovisno o parametrima za statičko i dinamičko trenje.

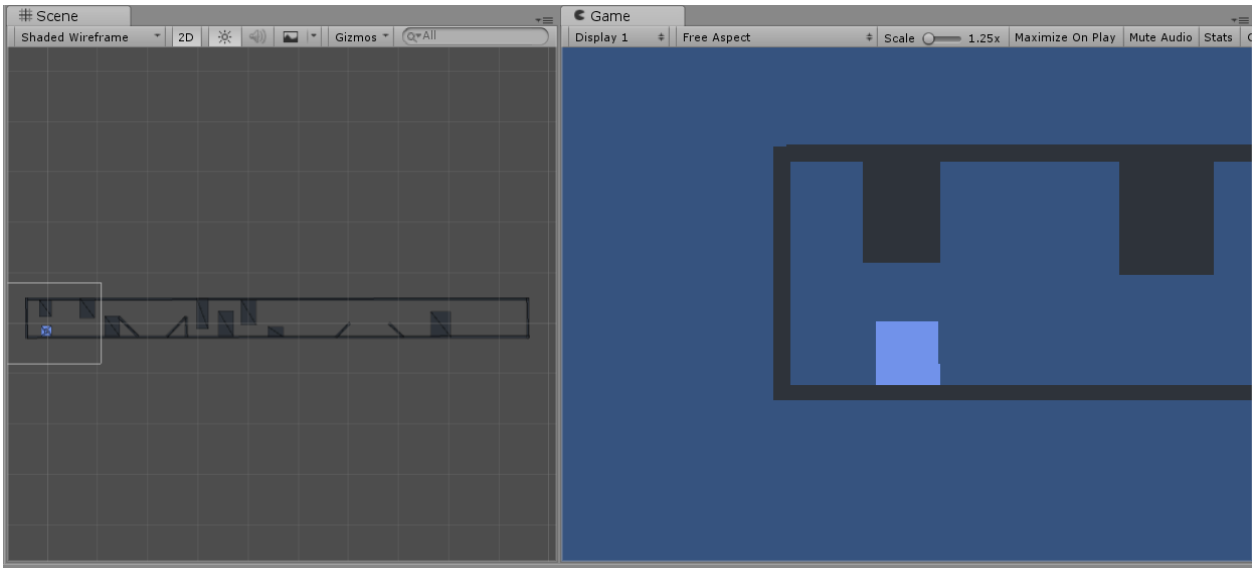
2.4 Vremenska integracija

Kako bi se deformacija ispravno prikazala potrebno ju je animirati uvođenjem vremenske dimenzije u sustav. Pri uvođenju vremenske integracije zbog svoje jednostavnosti i stabilnosti korištena je semi-implicitna Eulerova metoda te se pomoću nje izračuna pomak svih točaka za sljedeći vremenski trenutak.

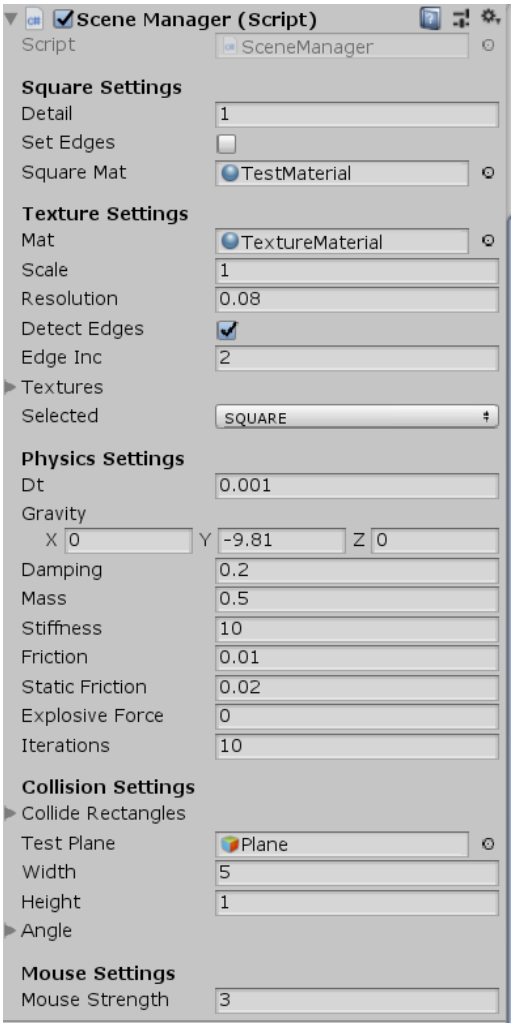
$$v(t+\Delta t) = v(t) + \Delta t\, F(v(t), x(t), t)$$
$$x(t+\Delta t) = x(t) + \Delta t\, v(t+\Delta t)$$

3. Upute za korištenje

Sustav sadrži dvije scene: *SampleScene* i *DemoScene*. *SampleScene* je inicijalna scena u kojoj se vršilo početno testiranje, a *DemoScene* je scena koja prije svega služi za demonstraciju sustava, ali se može koristiti i za testiranje (slika 5). U sceni se nalazi objekt *ObjectRenderer* koji sadrži skriptu *SceneManager* koja predstavlja glavnu skriptu za pokretanje sustava i preko koje se mijenjanjem i unošenjem različitih parametara mijenja stanje deformabilnog objekta. Postoje dva načina preko kojih se može stvoriti pravokutnik za kolizije. Prvi način je da se u *Editoru* doda položaj točke, visina, širina i kut te se na temelju tih parametara stvori pravokutnik, a drugi način je da se može kreirati novi *GameObject* u obliku pravokutnika sa tagom *CollisionObject.* Parametre koje je moguće promijeniti i unijeti vidljivi su u *Editoru* označavanjem *ObjectRenderer* objekta (slika 6), no potrebno je pažljivo mijenjati parametre pošto simulacija deformacije za određene vrijednosti neće biti stabilna.



Slika 5. Prikaz scene DemoScene



Slika 6. Prikaz parametara u Editoru

Square Settings

- Detail – razina detalja za kvadar, za cijeli broj *n* uniformno se unosi n^2 točaka unutar kvadra
- Set Edges – označavanjem se unose točke i na rubove kvadra
- Square Mat – materijal kvadra

Texture Settings

- Mat – materijal objekta stvorenog pomoću teksture
- Scale – razmjer objekta
- Resolution – određuje koliko će se točaka stvarati na teksturi, što je broj manji to će se više točaka stvoriti na teksturi
- Detect Edges – označavanjem će se detektirati rubovi na teksturi što znači da će model biti precizniji izgledu teksturi, ali će biti potrebno više vremena jer se izrađuje puno točaka
- Edge Inc – broj na temelju kojega se preskaču pikseli kako bi dobili manje točaka i kako bi se ubrzala triangulacija, ali će time biti smanjena preciznost

- Textures – niz tekstura koje određuju izgled objekta
- Selected – odabrani objekt, može biti: DONUT, ARMADILLO, BUNNY, SQUARE

Physics Settings

- Dt – vremenski razmak, što je niži, to će simulacija biti preciznija, ali će zahtijevati više procesorske snage.

- Gravity – sila gravitacije prikazana kao vektor
- Damping – konstanta prigušenja
- Mass – ukupna masa objekta, masa pojedine točke će biti mass / broj_točaka
- Stiffness – konstanta opruge, parametar koji određuje čvrstoću tijela
- Static Friction – koeficijent statičkog trenja
- Friction – koeficijent dinamičkog trenja
- Explosive Force – parametar koji dodaje brzinu u pozitivnom smjeru osi y kada se stisne tipka *space*

Collision Settings

- Collide Rectangles – niz položaja sudarajućih pravokutnika
- Test Plane – pravokutni objekt koji inicijalno služi za testiranje i koristi se u sceni *SampleScene*
- Width – širina pravokutnika
- Height – visina pravokutnika
- Angle – niz kutova pravokutnika

Mouse Strength

- Mouse Strength – jačina sile kojom će objekt biti pokrenut u smjeru pokazivača miša kada se klikne na njega

4. Literatura

[1] Z. Huangfu, L. Yan, X. Liu, 20.11.2013, An Improved Mass-spring Model for Simulation of Soft Tissue Deformation, Department of Information Engineering, North China University of Water Resources and Electric Power, China
[2] Y. Wang, Y. Xiong, K. Xu, K. Tan, G. Guo, A Mass-Spring Model for Surface Mesh Deformation Based on Shape Matching, School of Computer Science, National University of Defense Technology, Changsha, 410073, China, Center for Technique Education, PLA General Hospital, Beijing, 100853, China, Siječanj 2006.
[3] Nealen A., Müller M., Keiser R., Boxerman E., Carlson M., *Physically Based Deformable Models in Computer Graphics*, EuroGraphics 2005, Dublin, Republika Irska, 2005., stranice 1 – 6
[4] Torres L. M., *Fracture Modeling in Computer Graphics*, Universitat de Girona 2011., stranice 13-14
[5] Dinamika, Računalna grafika, Zagreb, FER, stranica 9
[6] Delaunay Triangulation, https://en.wikipedia.org/wiki/Delaunay_triangulation, 19.1.2019.