

Gerstnerovi valovi

Projekt za kolegij Računalna Animacija na Fakultetu Elektrotehnike i Računarstva u Zagrebu za akademsku godinu 25./26.

Sadržaj:

Upute za pokretanje	2
Pokretanje izgrađene izvršne datoteke.....	2
Izgradnja izvršne datoteke.....	2
Kontrole unutar aplikacije.....	2
Vizualizacija valova pomoću sjenčara	3
Interakcija s valovima na procesorskoj jedinici.....	5
Slike ekrana.....	7

Upute za pokretanje

Kao primjer vizualizacije Gerstnerovih valova i interakcije s njima, napravljena je jednostavna aplikacija u kojoj korisnik može upravljati brodom koji plovi po tim valovima i skuplja bačve (koje plutaju na valovima).

Pokretanje izgrađene izvršne datoteke

Pokretanje aplikacije može se napraviti preko već izgrađene izvršne datoteke koja se nalazi u komprimiranoj mapi:

`Waverider.zip`

Unutar ove mape nalazi se izvršna datoteka čijim pokretanjem će se pokrenuti aplikacija:

`Waverider.exe`

Izgradnja izvršne datoteke

Za samostalnu izgradnju izvršne datoteke potrebno je instalirati program UnityHub s kojim je moguće otvoriti projekt unutar Waverider mape.

Nakon toga, potrebno je (preko UnityHub programa) instalirati program Unity Editor 2022.3.62f2 te otvoriti projekt u njemu.

Unutar programa Unity Editor moguće je izgraditi izvršne datoteke aplikacije standardnim koracima.

Kontrole unutar aplikacije

Kontrole unutar aplikacije su sljedeće:

- W - davanje brzine unaprijed,
- S - davanje brzine unatrag,
- A - skretanje ulijevo,
- D - skretanje udesno,
- Mouse Right Click + Mouse Move - pomicanje orbitalne kamere,
- Mouse Scroll - promjena zoom-a kamere,
- Escape - gašenje aplikacije.

Vizualizacija valova pomoću sjenčara

Valovi se unutar sjenčara vrhova stvaraju pomicanjem vrhova primarno na y-osi za 3 različita Gerstnerova vala. Za svaki vrh se poziva sljedeća funkcija 3 puta (za svaki val jedna):

```
// Function which finds the world position of a vertex for one Gerstner wave.
void GerstnerPosition_float
(
    float3 world_pos,
    float height,
    float frequency,
    float speed,
    float peak,
    float3 direction,
    float TIME,
    out float3 Position
)
{
    float2 dir = normalize(direction.xz);

    float k = frequency;
    float w = k * speed;

    float phase = k * dot(dir, world_pos.xz) + w * TIME;

    float cosP = cos(phase);
    float sinP = sin(phase);

    // Horizontal displacement (XZ).
    float2 horizontal = peak * height * dir * cosP;

    // Vertical displacement.
    float vertical = height * sinP;

    Position = float3
    (
        world_pos.x + horizontal.x,
        vertical,
        world_pos.z + horizontal.y
    );
}
```

Svaki val za sebe ima 5 svojstava koja ga definiraju:

- height – Amplituda vala
- frequency – Frekvencija vala
- speed – Brzina kretanja vala
- peak – Skaliranje faktora na y-osi
- direction – Smjer vala

Za svaku funkciju, kao izlaz, dobiva se nova pozicija vrha. Zatim se za sve valove (u ovom slučaju 3) te pozicije zbrajaju da se dobije krajnja pozicija vrha u prostoru. Originalna visina vrha u originalnom koordinatnom sustavu se zanemaruje, ali se u završnom sjenčaru pribraja još i visina centra mreže kako bi se visina mogla kontrolirati.

Normale se računaju pomoću sljedeće funkcije:

```
// Function which finds the normal of a vertex for one Gerstner wave.
void GerstnerNormal_float
(
    float3 world_pos,
    float height,
    float frequency,
    float speed,
    float peak,
    float3 direction,
    float TIME,
    out float3 Normal
)
{
    float2 dir = normalize(direction.xz);

    float k = frequency;
    float w = k * speed;

    float phase = k * dot(dir, world_pos.xz) + w * TIME;

    float cosP = cos(phase);
    float sinP = sin(phase);

    // Partial derivatives (tangent vectors).
    float3 tangentX = float3(1, dir.x * height * k * cosP, 0) - peak * height * k
    * float3(dir.x * dir.x, 0, dir.x * dir.y) * sinP;
    float3 tangentZ = float3(0, dir.y * height * k * cosP, 1) - peak * height * k
    * float3(dir.x * dir.y, 0, dir.y * dir.y) * sinP;

    // Normal = cross product of tangent vectors.
    Normal = normalize(cross(tangentZ, tangentX));
}
```

U sjenčaru se sve normale (za sve valove) zbrajaju te normaliziraju, a rezultati dobro aproksimiraju stvarne normale za vrhove (u slučaju da zbroj svih varijabli peak za sve valove manje ili jednak 1).

Dodatno je napravljen i sjenčar fragmenata kako bi voda izgledala realističnije, ali se u tom sjenčaru ne rade stvari bitne za Gerstnerove valove.

Interakcija s valovima na procesorskoj jedinici

Za svaki objekt koji pluta ili plovi na površini vode/valova se za svako iscrtavanje ekrana izračunava aproksimacija visine valova u 4 točke:

- Naprijed i nazad kako bi se moglo izračunati okretanje u tom smjeru.
- Lijevo i desno kako bi se moglo izračunati okretanje u tom smjeru.

Za sve objekte koji plutaju ili plove na površini vode se, nakon što su izračunati novi Eulerovi kutovi, provodi rotacija kako bi kretanje po valovima bilo realistično.

Aproksimacija visine valova u jednoj točki izračunava se pomoću slijedeće funkcije:

```
public float GetHeightAt(Vector3 world_pos)
{
    // Converting world position to XZ plane for 2D sampling.
    Vector2 target_xz = new Vector2(world_pos.x, world_pos.z);
    Vector2 sample_xz = target_xz;

    float time = Time.time;
    const int iterations = 4;

    // Iteratively adjusting sample point for horizontal Gerstner displacement.
    for (int it = 0; it < iterations; it++)
    {
        Vector2 total_horizontal = Vector2.zero;

        for (int i = 0; i < _waves.Length; i++)
        {
            GerstnerWave w = _waves[i];

            Vector2 dir = new Vector2(w.Direction.x, w.Direction.z).normalized;

            float k = w.Frequency;
            float omega = k * w.Speed;

            float phase = k * Vector2.Dot(dir, sample_xz) + omega * time;

            float cos_p = Mathf.Cos(phase);

            total_horizontal += (w.Peak * w.Height * dir * cos_p) / _waves.Length;
        }

        // Pulling sample points back toward original surface position.
        sample_xz = target_xz - total_horizontal;
    }

    // Computing final height at the corrected position.
    float height = 0f;

    for (int i = 0; i < _waves.Length; i++)
    {
        GerstnerWave w = _waves[i];

        Vector2 dir = new Vector2(w.Direction.x, w.Direction.z).normalized;
```

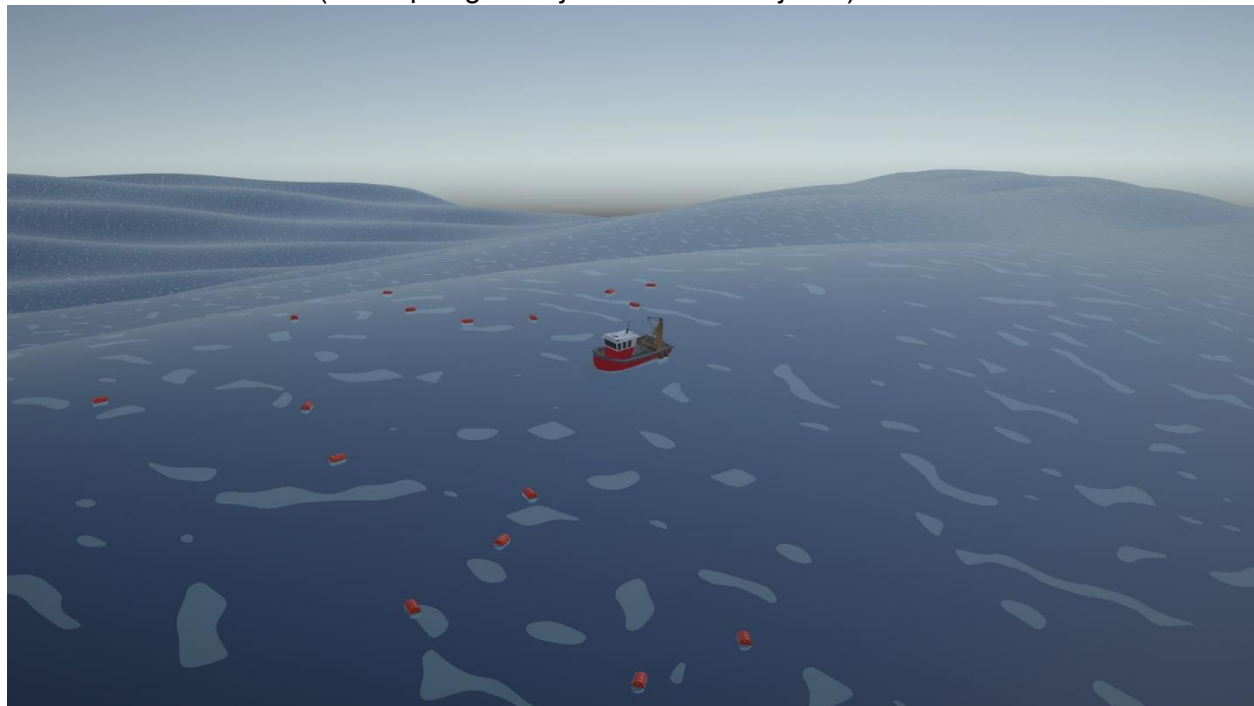
```
float k = w.Frequency;  
float omega = k * w.Speed;  
  
float phase = k * Vector2.Dot(dir, sample_xz) + omega * time;  
height += w.Height * Mathf.Sin(phase);  
}  
return height;  
}
```

Prikaz točaka i osi na čije rotacije točke utječu vidljiv je na idućoj slici (na y-os utječe igrač):

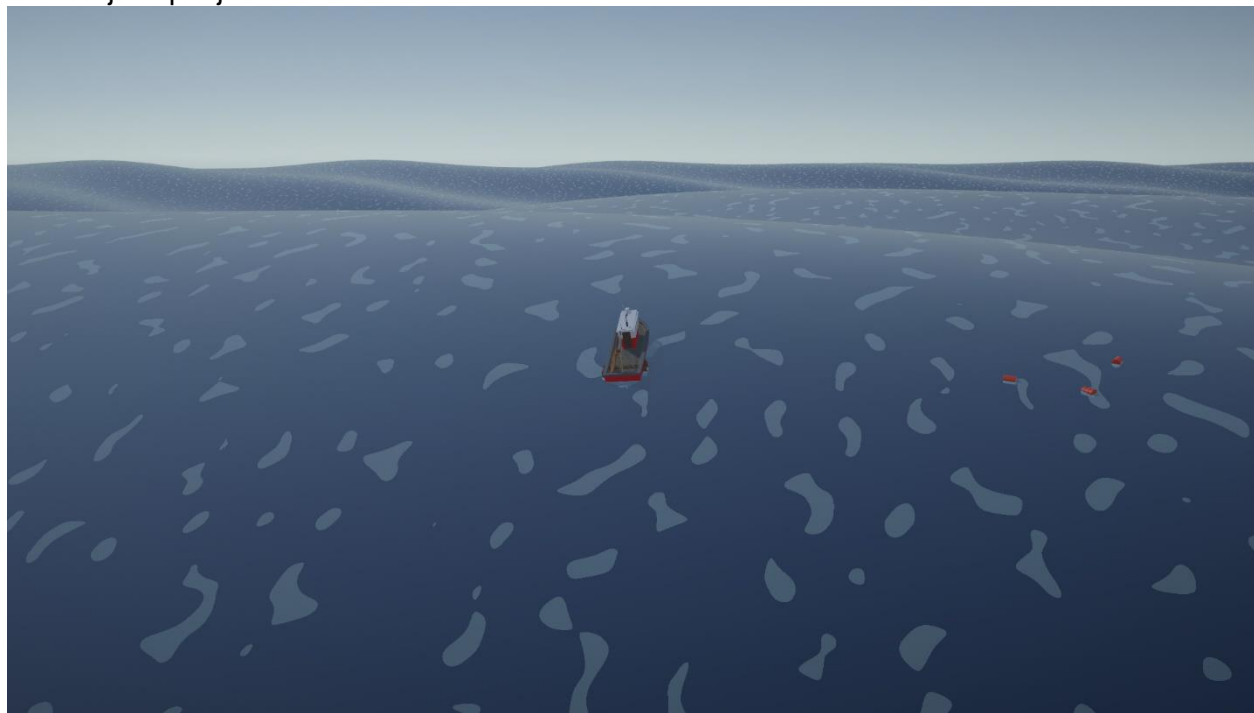


Slike ekrana

Brod okružen bačvama (svi se prilagođavaju valovima u dvije osi):



Brod koji se penje uz val:



Brod koji plovi uz val tako da se naginje na bok:

