PACKET TERROR

Proiect realizat de:

Popescu Vlad

Pescariu Matei

Duzi Mihai

Ideea Proiectului

Un joc Tower Defense cu o temă bazată pe Cisco Packet Tracer în care jucătorii protejează marele intranet sovietic de atacuri inamice.

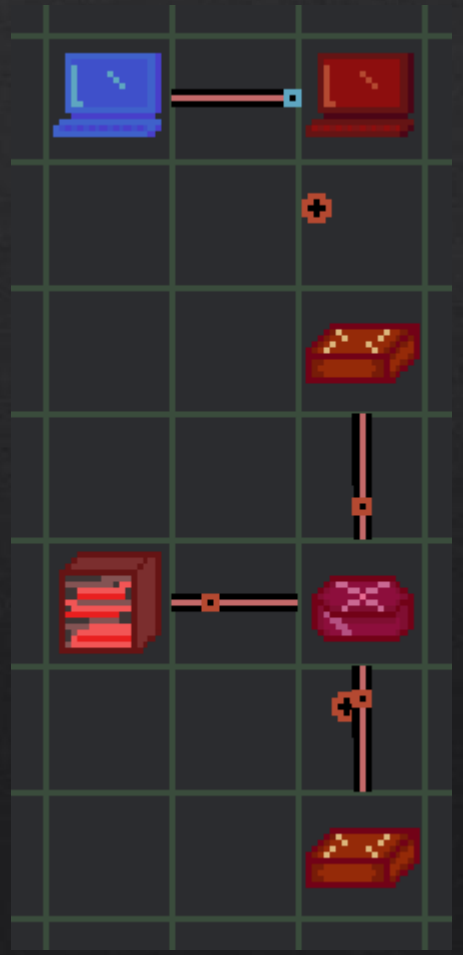
Jucătorul are un PC care este atacat de pachete emise de PC-ul inamic.

Poate cumpăra și plasa servere ca să-și emită propriile pachete, cabluri pe care acestea se deplaseaza, routere pentru a le directiona spre alt dispozitiv, switch-uri care lansează pachete spre pachetele inamicului, distrugându-le.

Toate aceste dispozitive pot fi upgradate, iar jocul va avea mai multe nivele cu configurații de început și wave-uri de inamici diferite.

Backlogul proiectului: aici

Demo: aici



Arhitectura

Proiectul este realizat în Bevy, un game engine scris în Rust bazat pe design pattern ECS (Entity Component System) ce prioritizează viteza de compilare și paralelizarea codului într-un mediu non-OOP.

Este complet multi-platform, cu build-uri iterative, crate-uri (librării) open-source si extrem de diferit de alte engine-uri precum Unity, Unreal, Godot, etc.

Fiind relativ nou, bevy nu are un editor oficial și multe alte feature-uri comune în alte game engine-uri, în schimb este foarte modular, avem acces low-level, putem modifica tot codul din spate, și este backed up de o selecție mare de crate-uri open-source ce îi adauga diferite feature-uri.

Aspecte tehnice

ECS (Entity Component System) este bazat pe crearea unor entități formate din tupluri de componente (ce sunt la rândul lor structuri, adică tupluri de date) si manipularea lor din sisteme.

Aceste sisteme sunt declarate la un nivel global și accesează pool-uri de entități în același timp, față de engine-urile OOP care au comportamentul obiectelor definit direct pe componente.

Aceasta arhitectura este extrem de eficienta pentru ca [aproape] toate sistemele rulează în paralel, suportă builduri iterative, iar componentele sunt stocate și accesate eficient din anumite structuri speciale.

De exemplu, pe următorul slide avem un sistem care rulează în fiecare frame, accesează toate switch-urile (entitățile cu componenta 'Switch'), plus alte resurse necesare, și spawnează proiectile pentru fiecare pachet care este pozitionat deasupra lui.

```rust
fn shoot_projectiles(
    player_packets: Query<(Entity, &Transform), With<PlayerPacket>>,
    enemy_packets: Query<(Entity, &Transform), With<EnemyPacket>>,
    mut commands: Commands,
    asset_server: Res<AssetServer>,
    switches: Query<(&GlobalTransform, &Switch, &ProjectileType)>,
    grid: ResMut<Grid>,
) {
    for (packet_entity, pos) in &player_packets {
        if let Some((t_switch, _, &projectile_type)) = grid
            .get_element(pos.translation.truncate())
            .and_then(|e| switches.get(e).ok())
        {
            if let Some((target, _)) = enemy_packets.iter().max_by(|&(_, t1), &(_, t2)| {
                t1.translation
                    .distance(pos.translation)
                    .total_cmp(&t2.translation.distance(pos.translation))
                    .reverse()
            }) {
                commands.spawn((
                    Projectile {
                        target,
                        projectile_type,
                    },
                    Sprite::from_image(asset_server.load("projectile.png")),
                    Transform::from_translation(t_switch.translation()),
                ));
            }

            commands.entity(packet_entity).despawn();
        }
    }
}
```

Pentru prezentarea finală...

Upgrade-uri pentru server/switch/router

Îmbunătățiri pentru UI

Efecte speciale

Nivele

Vă mulțumim pentru atenție!