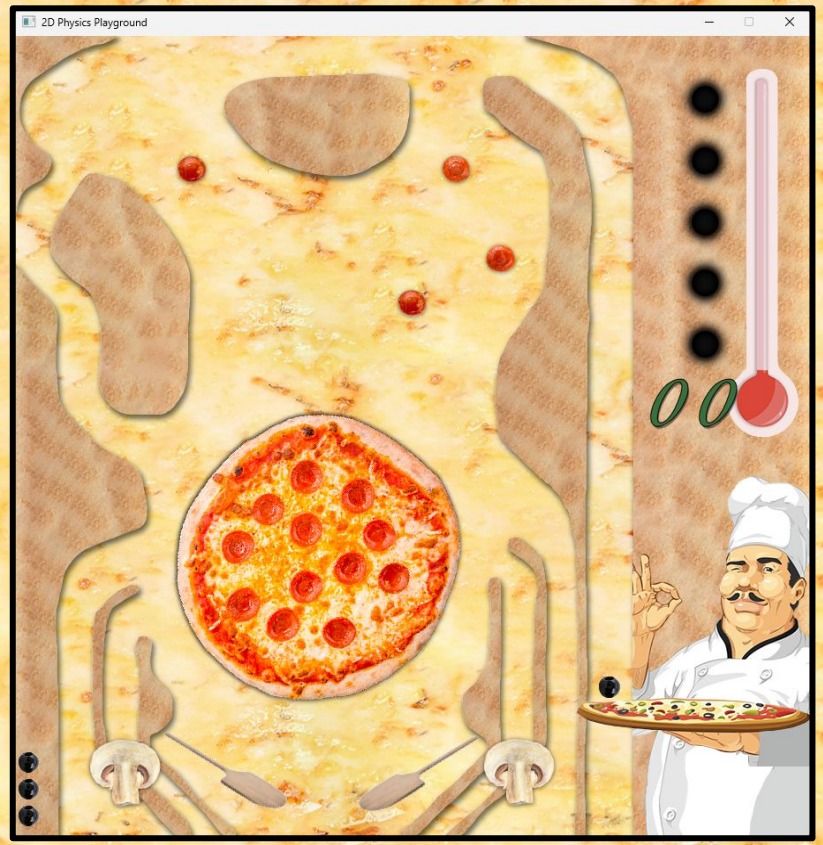
Caracoles Cansaus

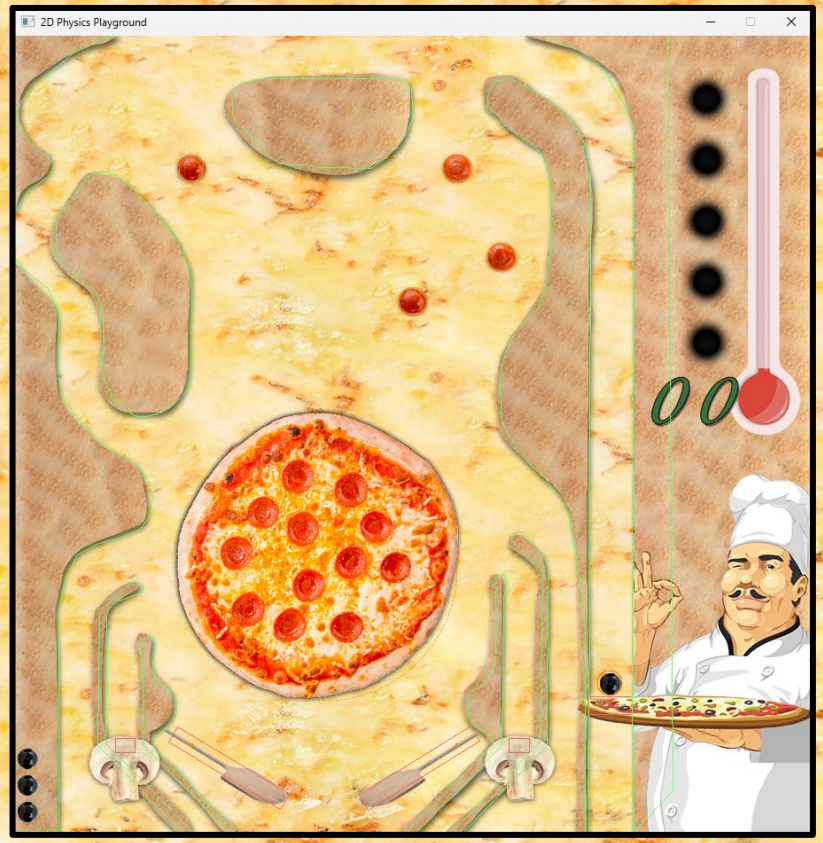
Jiajie Lin
Guillem Montes
Raül Sanchez
Martí Sabaté

PANTALLAS


2D Physics Playground
La Capricciosa
Nuovo Gioco
Esci


2D Physics Playground
0 0

PANTALLAS


2D Physics Playground


2D Physics Playground
Fine del Gioco
Pulsa espacio para continuar

Debug

Controles

```
## Keyboard Controls
### Basic Controls
 - DOWN ARROW - Compress/Release spring
 - LEFT ARROW - Move the left flipper
 - RIGHT ARROW - Move the right flipper
 - SPACE - Confirm/Start Game
 - P - Reset Ball
 - ESC - Close Game


### Debug Controls

 - F1 - Toggle Debug mode
 - F2 - Increase gravity scaler
 - F2 + LEFT SHIFT - Increase gravity scaler * 10
 - F3 - Decrease gravity scaler
 - F3 + LEFT SHIFT - Decrease gravity scaler * 10
 - F4 - Increase bouncing factor
 - F5 - Decrease bouncing factor
 - F6 - Increase life in 1 (max:3)
 - LEFT_CLICK On ball with Debug mode - Active the mouse joint
```

```
if (App->input->GetKey(SDL_SCANCODE_F6) == KEY_DOWN) {
    vidas++;
    vidas = min(vidas, 3);
}
```

```
if (App->input->GetKey(SDL_SCANCODE_F2) == KEY_DOWN) {
    if (App->input->GetKey(SDL_SCANCODE_LSHIFT) == KEY_REPEAT) {
        gravityScale += 1;
    }
    else {
        gravityScale += 0.1f;
    }

}
if (App->input->GetKey(SDL_SCANCODE_F3) == KEY_DOWN) {
    if (App->input->GetKey(SDL_SCANCODE_LSHIFT) == KEY_REPEAT) {
        gravityScale -= 1;
    }
    else {
        gravityScale -= 0.1f;
    }
}
gravityScale = max(gravityScale, 0);


if (App->input->GetKey(SDL_SCANCODE_F4) == KEY_DOWN) {

    bounceCoefficient += 0.1f;


}
if (App->input->GetKey(SDL_SCANCODE_F5) == KEY_DOWN) {

    bounceCoefficient -= 0.1f;

}
bounceCoefficient = max(bounceCoefficient, 0);
bounceCoefficient = min(bounceCoefficient, 1);
```

```
ball->body->SetGravityScale(gravityScale);
ball->body->GetFixtureList()[0].SetRestitution(bounceCoefficient);
```

Debug
Controles

```
if (App->input->GetMouseButton(SDL_BUTTON_LEFT) == KEY_DOWN)
{
    // test if the current body contains mouse position
    b2Vec2 p = { PIXEL_TO_METERS(App->input->GetMouseX()), PIXEL_TO_METERS(App->input->GetMouseY()) };
    if (f->GetShape()->TestPoint(b->GetTransform(), p) == true)
    {

        mouse_body = b;


        b2Vec2 mousePosition;
        mousePosition.x = p.x;
        mousePosition.y = p.y;

        b2MouseJointDef def;
        def.bodyA = ground;
        def.bodyB = mouse_body;
        def.target = mousePosition;
        def.dampingRatio = 0.9f;
        def.frequencyHz = 5.0f;
        def.maxForce = 200.0f * mouse_body->GetMass();


        mouse_joint = (b2MouseJoint*)world->CreateJoint(&def);
    }
}
```

```
//Mantener click
if (mouse_body != nullptr && mouse_joint != nullptr)
{
    if (App->input->GetMouseButton(SDL_BUTTON_LEFT) == KEY_REPEAT)
    {

        b2Vec2 mousePosition;
        mousePosition.x = PIXEL_TO_METERS(App->input->GetMouseX());
        mousePosition.y = PIXEL_TO_METERS(App->input->GetMouseY());
        mouse_joint->SetTarget(mousePosition);


        App->renderer->DrawLine(METERS_TO_PIXELS(mouse_body->GetPosition().x), METERS_TO_PIXELS(mouse_body->GetPosition().y), App->input->GetMouseX(), App->input->GetMouseY(), 255, 0, 0);
    }
}

//Soltar click
if (mouse_body != nullptr && mouse_joint != nullptr)
{
    if (App->input->GetMouseButton(SDL_BUTTON_LEFT) == KEY_UP)
    {

        world->DestroyJoint(mouse_joint);


        mouse_joint = nullptr;
        mouse_body = nullptr;
    }
}
```

Física

Bola

```
ball = App->physics->CreateCircle(155, 700, 15);
ball->listener = this;
ball->type = BALL;
```

```
void ModuleSceneIntro::OnCollision(PhysBody* bodyA, PhysBody* bodyB)
{
    int x, y;

    App->audio->PlayFx(bonus_fx);


    if ((bodyA->type == BALL && bodyB->type == RESETBALL) || (bodyA->type == RESETBALL  && bodyB->type == BALL)) {
        crearBola = true;
    }
    if ((bodyA->type == BALL && bodyB->type == PLATAFORMA_ROTANTE)) {
        estaRotando = true;
    }
    if (bodyA->type == BALL && bodyB->type == REBOTADOR) {
        puntuacionJuego += puntuacionAlTocar;
    }
    if (bodyA->type == BALL && bodyB->type == REBOTADOR_DE_MUELLE) {
        ball->body->ApplyLinearImpulse(b2Vec2(0, -20), ball->body->GetWorldCenter(), true);
    }
    if (bodyA->type == BALL && bodyB->type == REBOTADOR_DE_MUELLE2) {
        ball->body->ApplyLinearImpulse(b2Vec2(0, -12), ball->body->GetWorldCenter(), true);
    }


}
```

Física

Muelle

```
//Muelle inicio
muelleInicio = App->physics->CreateRectangle(673, 775, 50, 20);

muelleInicio->body->SetFixedRotation(true);
muelleInicioPoint = App->physics->CreateCircle(673, 830, 2);
muelleInicioPoint->body->SetType(b2_staticBody);

b2DistanceJointDef muelleDef;

muelleDef.bodyA = muelleInicio->body;
muelleDef.bodyB = muelleInicioPoint->body;

muelleDef.localAnchorA.Set(0, 0);
muelleDef.localAnchorB.Set(0, 0);

muelleDef.length = 1.5f;

muelleDef.collideConnected = true;

muelleDef.frequencyHz = 7.0f;
muelleDef.dampingRatio = 0.05f;

b2PrismaticJoint* muelleJoint = (b2PrismaticJoint*)App->physics->GetWorld()->CreateJoint(&muelleDef);
```

```
if (App->input->GetKey(SDL_SCANCODE_DOWN) == KEY_REPEAT) {
    //paletaInicio->body->ApplyForceToCenter(b2Vec2(0, fuerzaPaleta), 1);


    if (springForce < 900) {
        springForce += 10;

    }
    muelleInicio->body->ApplyForceToCenter(b2Vec2(0, springForce), 1);
}

if (App->input->GetKey(SDL_SCANCODE_DOWN) == KEY_UP) {
    springForce = 0;
}
```

Física

Paletas

```
if (App->input->GetKey(SDL_SCANCODE_LEFT) == KEY_REPEAT) {
    paletaIzquierdo->body->ApplyForceToCenter(b2Vec2(0, fuerzaPaleta), 1);
}
if (App->input->GetKey(SDL_SCANCODE_RIGHT) == KEY_REPEAT) {
    paletaDerecho->body->ApplyForceToCenter(b2Vec2(0, fuerzaPaleta), 1);
}
```

```
paletaIzquierdo = App->physics->CreateRectangle(x1, y1, w, h);
paletaIzquierdoPoint = App->physics->CreateCircle(x1, y2, 2);
paletaIzquierdoPoint->body->SetType(b2_staticBody);


b2RevoluteJointDef paletaIzquierdoJoint;

paletaIzquierdoJoint.bodyA = paletaIzquierdo->body;
paletaIzquierdoJoint.bodyB = paletaIzquierdoPoint->body;
paletaIzquierdoJoint.referenceAngle = 0 * DEGTORAD;
paletaIzquierdoJoint.enableLimit = true;
paletaIzquierdoJoint.lowerAngle = -30 * DEGTORAD;
paletaIzquierdoJoint.upperAngle = 30 * DEGTORAD;
paletaIzquierdoJoint.localAnchorA.Set(PIXEL_TO_METERS(-50), 0);
paletaIzquierdoJoint.localAnchorB.Set(0, 0);
b2RevoluteJoint* joint_leftFlipper = (b2RevoluteJoint*)App->physics->GetWorld()->CreateJoint(&paletaIzquierdoJoint);
```

Física

Rebotes

```
bola = App->physics->CreateBolas(200, 150, 14);
bola->body->SetType(b2_staticBody);
bola->body->SetFixedRotation(true);
bola->type = REBOTADOR;



bola2 = App->physics->CreateBolas(500, 150, 16);
bola2->body->SetType(b2_staticBody);
bola2->body->SetFixedRotation(true);
bola2->type = REBOTADOR;

bola3 = App->physics->CreateBolas(550, 250, 16);
bola3->body->SetType(b2_staticBody);
bola3->body->SetFixedRotation(true);
bola3->type = REBOTADOR;

bola4 = App->physics->CreateBolas(450, 300, 16);
bola4->body->SetType(b2_staticBody);
bola4->body->SetFixedRotation(true);
bola4->type = REBOTADOR;
```

```
PhysBody* ModulePhysics::CreateBolas( int x, int y, int radius)
{
    b2BodyDef body;
    body.type = b2_dynamicBody;
    body.position.Set(PIXEL_TO_METERS(x), PIXEL_TO_METERS(y));

    b2Body* b = world->CreateBody(&body);

    b2CircleShape shape;
    shape.m_radius = PIXEL_TO_METERS(radius);
    b2FixtureDef fixture;
    fixture.shape = &shape;
    fixture.density = 1.0f;
    fixture.restitution = 0.7f;


    b->CreateFixture(&fixture);

    PhysBody* pbody = new PhysBody();
    pbody->body = b;
    b->SetUserData(pbody);
    pbody->width = pbody->height = radius;

    return pbody;


}
```

Física

Pizza

```
if (posX <= ruletaX) {
    if (posY <= ruletaY) {
        //Esquina arriba izquierda
        ball->body->ApplyForceToCenter(b2Vec2(ruletaForce, 0), true);

    }
    else {
        //Esquina abajo izquierda
        ball->body->ApplyForceToCenter(b2Vec2(0, -ruletaForce), true);
    }
}
else {
    if (posY <= ruletaY) {
        //Esquina arriba derecha
        ball->body->ApplyForceToCenter(b2Vec2(0, ruletaForce), true);
    }
    else {
        //Esquina abajo derecha
        ball->body->ApplyForceToCenter(b2Vec2(-ruletaForce, 0), true);
    }
}
```