




A Golf Ball's Fantasy



CS 174A Group Project Final Demo
Group Member: JJ Choon, Aiqi You, Joanne Qiu



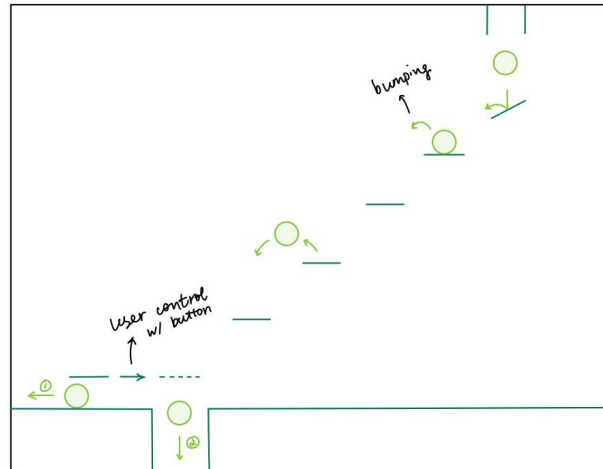
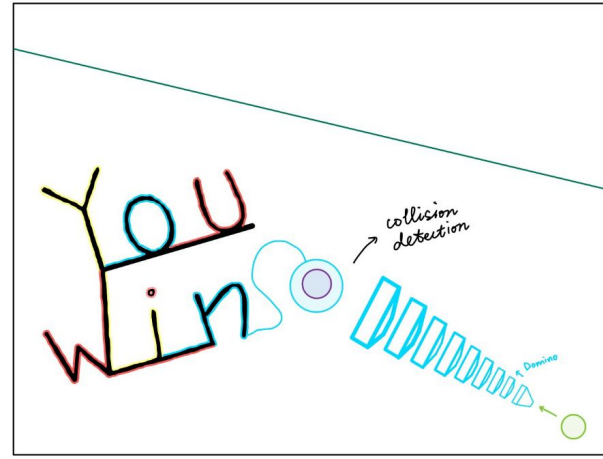
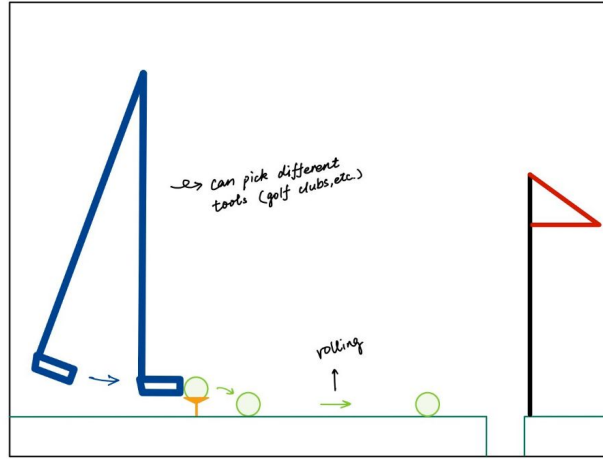
Contents

- Design
- Implementation
- Demo



Design

- 4 scenes depicting the adventure of a small golf ball:
 - Scene 1: Player swings the golf club and pushes the ball into the hole
 - Scene 2: The ball bounces on the small platforms and player needs to control the speed of the ball to land on the ground
 - Scene 3: If the ball successfully lands on the ground, it rolls towards dominoes, and the last domino will push the button, which triggers the sign "You Win"
 - Scene 4: If the ball doesn't land on the ground, it will drop into a water tank, and game over



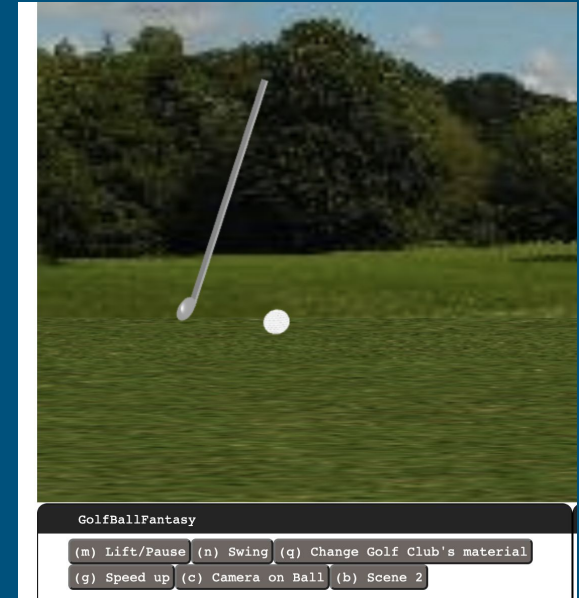
Implementation

- User control
- Collision detection
- Physics-based simulation
- Camera Control



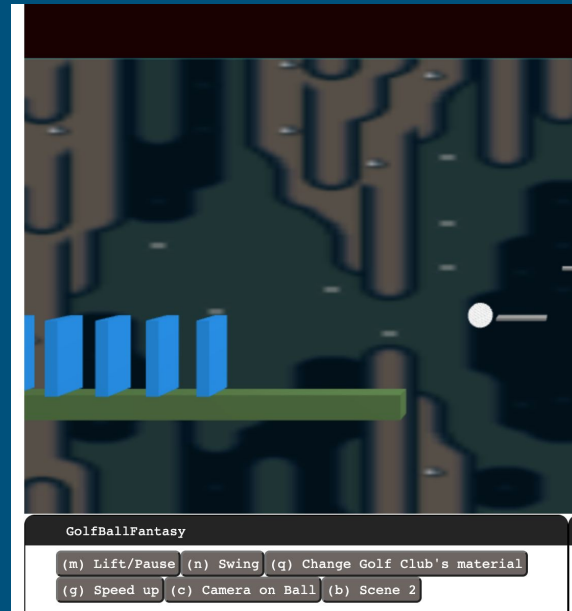
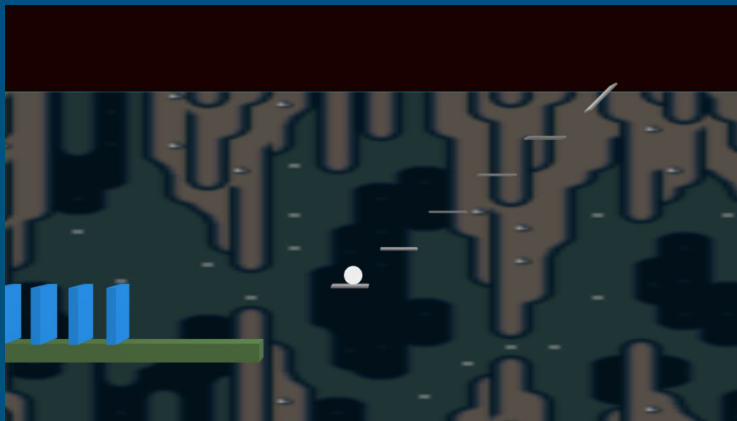
Implementation - Scene 1

- Features:
 - Customized golf club's materials
 - User control on golf club
 - Collision Detection
 - Pendulum simulation



Implementation - Scene 2

- Features:
 - Bouncing simulation
 - Collision Detection
 - User control on ball's speed



Implementation - Scene 2 Golf ball Motion

Constant acceleration motion

$$v_x = v_{x0} + a_x \Delta t$$

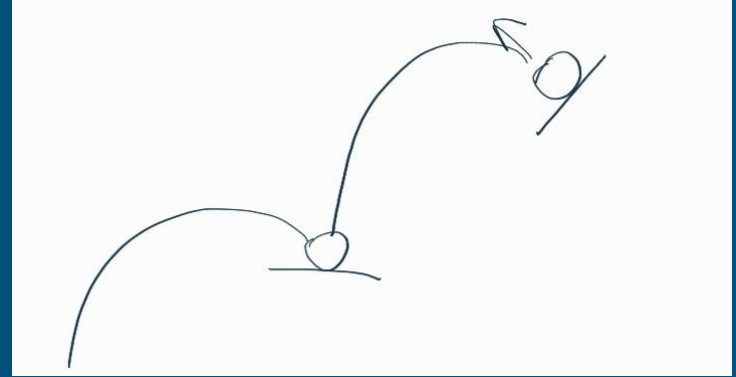
$$v_y = v_{y0} + a_y \Delta t$$

$$\Delta x = \frac{v_x + v_{x0}}{2} \Delta t$$

$$\Delta y = \frac{v_y + v_{y0}}{2} \Delta t$$

We prepend this delta-translation to the golf_ball transform matrix to set the ball in motion

Bouncing off the first platform



We set the x and y components of the bounce-off speed to be $1/(2\sqrt{2})$ of the initial speed to model collision and kinetic energy loss

Implementation - Scene 2 Collection Detection

For the first platform, we test whether the bottom right point of the ball hits the platform. For the other platforms, we test the golf ball bottom point.

On-Line Test

P is on P_1P_2 means $\frac{x-x_1}{y-y_1} = \frac{x_2-x_1}{y_2-y_1}$

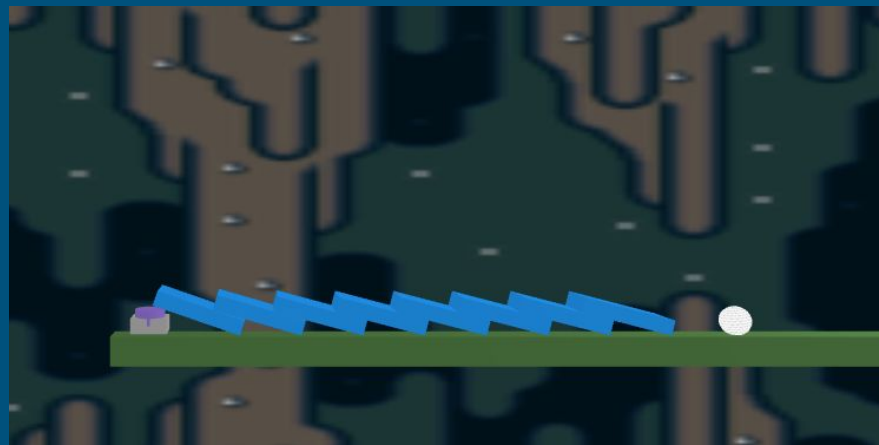
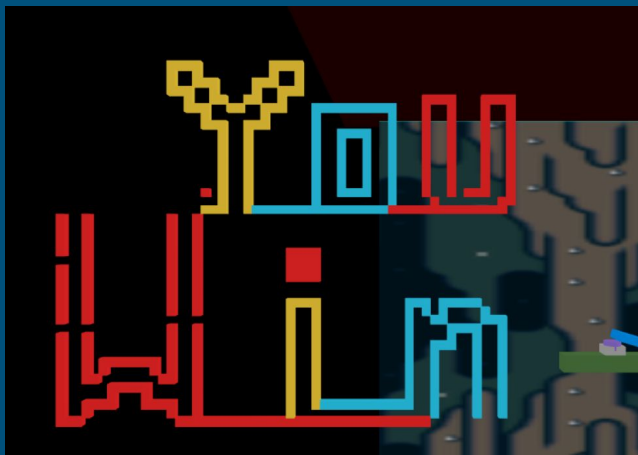
If $T_{1,2}(P) = (x-x_1)(y_2-y_1) - (x_2-x_1)(y-y_1)$

if +ve, P is on the right; if -ve, P is on the left



Implementation - Scene 3

- Features:
 - Collision detection
 - Top right edge with the right side of the next domino
 - Top right corner of the button with the left side of the last domino
 - Dominoes falling simulation



Implementation - Scene 3 Domino Motion

Falling domino:

Assume the domino is like a thin rod, then its moment of inertia $I = \frac{1}{3}mh^2$

When falling, $I\alpha = \tau = Fr = (mg \sin \theta)(\frac{1}{2}h) = \frac{1}{2}mgh \sin \theta$

So $\alpha = \frac{mgh \sin \theta}{\frac{1}{3}mh^2}$

$\alpha = \frac{mgh \sin \theta}{\frac{1}{3}mh^2} = \frac{3g}{2h} \sin \theta$

where θ is the rotation angle.

For Δt , we can approximate with

$$\omega = \int \alpha dt \approx \omega_0 + \alpha \Delta t$$

$$\theta = \int \omega dt \approx \theta_0 + \omega \Delta t$$

On hitting by the ball:



kinetic
Assume all the \hat{v} energy of the ball is transferred to the domino:

$$E_f = E_b$$

$$\frac{1}{2}I\omega^2 = \frac{1}{2}m_b v_b^2$$

$$\omega = \left[\frac{m_b v_b^2}{I} \right]^{1/2} = v_b \left[\frac{m_b v_b}{I} \right]^{1/2} = v_b \left[\frac{m_b v_b}{\frac{1}{3}m_d h^2} \right]^{1/2}$$

initial angular velocity of the 1st domino

$$\omega = \frac{v_b}{hd} \sqrt{\frac{3m_b v_b}{m_d}}, \text{ where } b \text{ is the ball and } d \text{ is the domino.}$$

*in the actual implementation we decided to let the ball keep some kinetic energy and bounce back

On hitting by another domino:

Assume the two dominoes move together

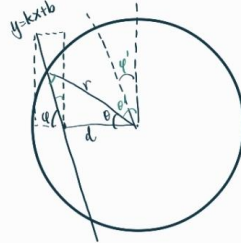


$$\frac{h}{\sin(\theta + \psi)} = \frac{d}{\sin \psi}, \quad \psi + \theta = \theta' \Rightarrow \psi = \theta' - \theta$$

$$\text{So } \frac{h}{\cos \theta} = \frac{d}{\sin(\theta' - \theta)}$$

$$d \cos \theta = h \sin(\theta' - \theta)$$

$$\frac{d}{dt}(d \cos \theta) = \frac{d}{dt}[h \sin(\theta' - \theta)]$$

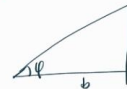


$$\frac{\sin(\psi + \frac{\pi}{2})}{r} = \frac{\sin(\theta' - \psi)}{d}$$

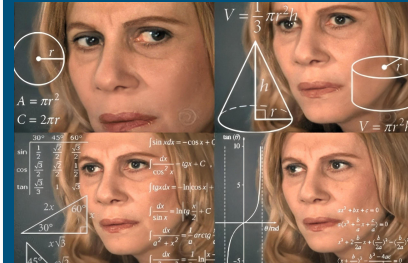
$$\frac{\cos \psi}{r} = \frac{\sin(\theta' - \psi)}{d}$$

$$\frac{d}{r} \cos \psi = \sin(\theta' - \psi)$$

$$\theta' = \psi + \arcsin\left(\frac{d}{r} \cos \psi\right)$$



$$\sin \phi = \frac{a}{b}$$

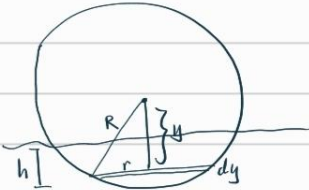
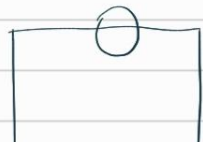


Implementation - Scene 4

- Features:
 - Collision detection
 - Modeling buoyancy
 - Splashing effect



Implementation - Scene 4 Buoyancy


$$r^2 = (R^2 - y^2)$$
$$V = \int_{-R}^{-R+h} \pi r^2 dy$$
$$= \int_{-R}^{-R+h} \pi (R^2 - y^2) dy$$
$$= \pi R^2 y - \frac{\pi}{3} y^3 \Big|_{-R}^{-R+h} = \pi \left(R h^2 - \frac{h^3}{3} \right)$$

$$F_b = \rho g V,$$
$$a = \frac{1}{m} (F_b - F_g) = \frac{1}{m} (F_b - mg) = \frac{F_b}{m} - g$$

```
const g = 9.8, liquid_density = 2000, radius = 0.024, mass = 0.046;  
const h_real = radius/1 * h;  
const immersed_volume = Math.PI*(radius*h_real*h_real - h_real**3/3);  
const buoyancy = liquid_density*g*immersed_volume;  
const acc = buoyancy / mass - g;  
this.golf_ball_acceleration.y = acc;
```

Camera Control

- 2 different camera angles on the golf ball (close up and far).
- Scene 2 camera angle.
- When “You Win”, automatically set the camera look at it.

Future Improvements

- Add friction to scene1 when the golf ball is rolling
- Add shadowing for the ball and use bump mapping for the water
- Add reset button for the golf ball

Thanks for listening!

Now, goes to

Demo
