Atom Collision*

1st 111511234 Yan-Tao, Chu
UEE (of Aff.)
NYCU (of Aff.)
Hsinchu, Taiwan
dctime.ee11@nycu.edu.tw

2nd 110550040 Ting-lin,Chu
DCP (of Aff.)
NYCU (of Aff.)
Hsinchu, Taiwan
runchu2002.cs10@nycu.edu.tw

***Abstract*—This document is an introduction to our game, "Atom Collision" .**

I. Introduction

"Atom Collision" is an innovative game that combines character customization and realistic physics. Developed with pygame and numpy, it allows players to create and personalize their own atoms(cars), witnessing accurate collisions and reactions. The game offers a seamless experience with intuitive controls, stunning visuals, and challenging scenarios. Whether for education or entertainment, "Atom Collision" provides an immersive adventure into the microscopic world, catering to players of all skill levels.(Referencing ChatGPT [1])

II. Usage

(1) Download "atom_collision.zip" in our repo.
(2) https://github.com/dctime/atom-collision/releases/tag/v1.0.0
(3) Unzip "atom_collision.zip" to wherever you like(Unzip it to a new folder is recommended).
(4) Execute "atom_collision.exe" and have fun!

III. Details

A. Rules

There are 2 players controlled by the same host(i.e. your PC).

First, both players need to take turns building their cars. Each car is composed of 1 core block and many wood/stone blocks.

After building both cars, the battle will start immediately. Player1 and player2 can control their cars with WASD/↑←↓→, respectively. Move carefully because there will be a gravity field centered at the center of the screen, of which strength will increase as time goes on. Also, once a block is disconnected to its core block, it will be destroyed right away.

The goal is to break the opponents car. Once one of the car's core is destroyed, then the game is over.

B. Gameplay

First, we'll need to build our cars. (Fig.1)

Press "Enter" to build the next car. When the second car is also built, press "Enter" to start the battle.(Fig.2)

With the game proceeding, the gravity field will be strengthened and its color will change, too.(Fig.3,4,5)

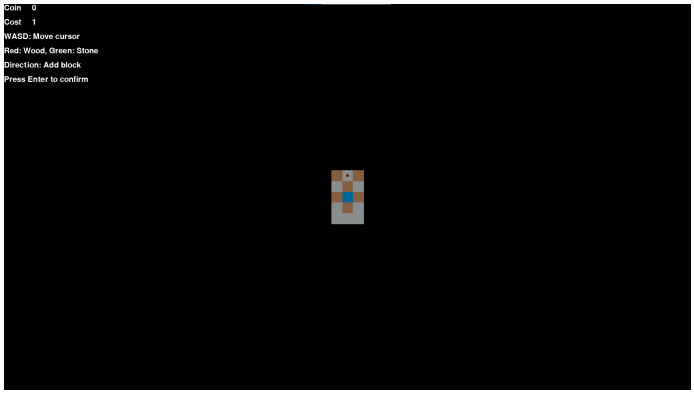

Fig. 1. Building cars.

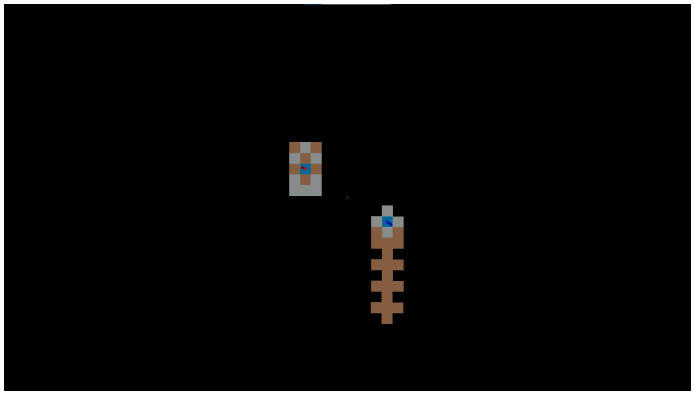

Fig. 2. Start battle.

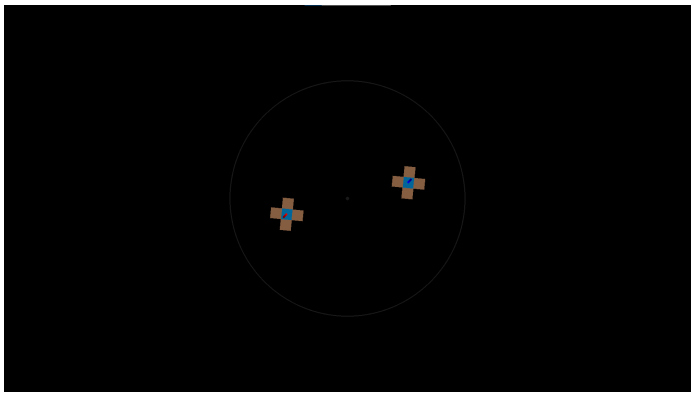

Fig. 3. Stage 1.

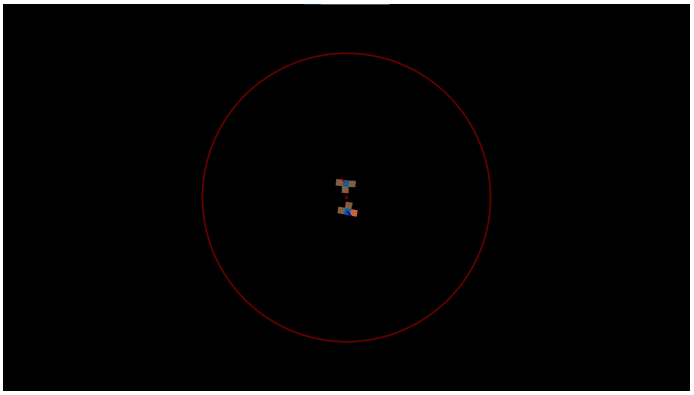

Fig. 4. Stage 2.

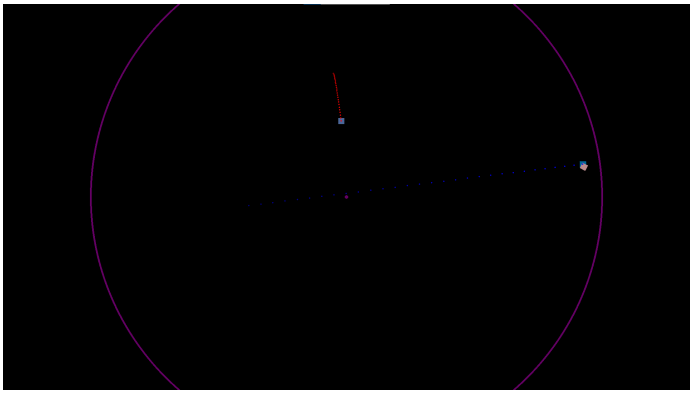

Fig. 5. Stage 3.

When the game ends, the screen will be paused, displaying the winner and the scoreboard.(Fig.6)

Fig. 6. Result.

Now, if you want to start a new game, press “Enter”; if you want to quit, press “Alt”+”F4”;(.(Fig.7)

C. UML Class Diagram

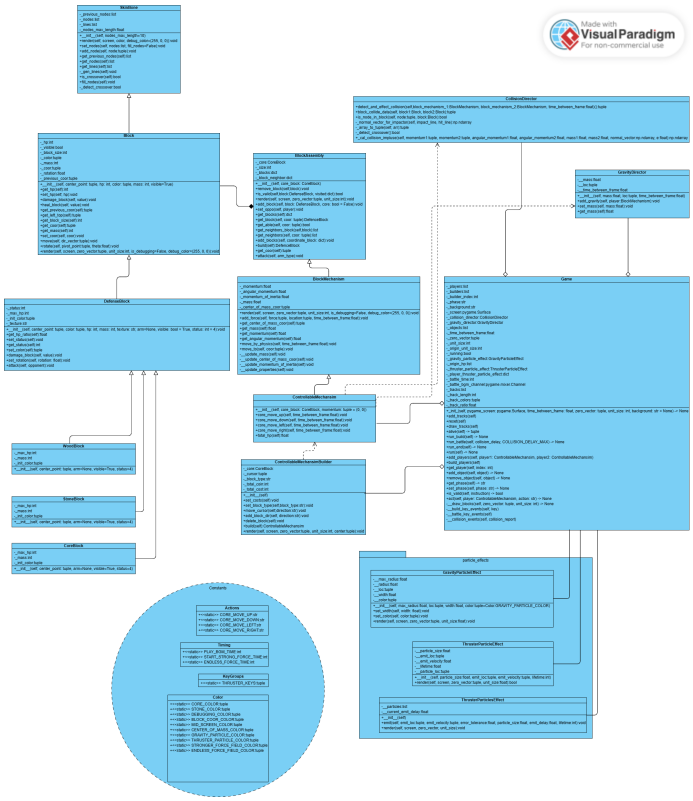

Fig. 7. UML Class Diagram.

The ”Game” class is the class we want to instantiate in the main.py, which is the python file we run when running the game. The ”Game” class has two cars, which are called ”ControllableMechanism”, fighting each other on the playground. These cars can be built by the ”ControllableMechanismBuilder”, which is been instantiated before the build phase starts. ”Game” also has ”CollisionDirector”, which checks and reacts when two cars collide with each other and ”GravityDirector” applies gravitational force to both cars.

Particle effects are part of the ”Game”. When the ”Game” wants to have particle effects, it needs to instantiate. Then, ”Game” needs to call ”render” member function in order to show it on the screen.

According to the diagram, ”ControllableMechanism” inherits ”BlockMechanism”, while ”BlockMechanism” inherits ”BlockAssembly”. ”BlockAssembly” is made of blocks, which it will be explained after a while, and it only contains the information of every block made of it and its relative positions from the core. ”BlockMechanism” contains the information related to movement, such as momentum, angular momentum. There is also rotate and move function to use in the class. Lastly, ”ControllableMechansim” contains information related to the controls of the car, such as telling core to emit flame, which causes the car to accelerate.

Notice that ”Block” inherits ”SkinBone”. ”SkinBone” can be any shape by define the points in order. ”Block” objects

are all squares, contains everything related to squares. "DefenseBlock" is a type of "Block", which can only do damage by collision. There are three types of blocks in the game: "CoreBlock", "WoodBlock", and "StoneBlock". "CoreBlock is the most important block in a car. If it's destroyed, then the game ends. It has the most mass and has the most hp in all the blocks. "StoneBlock" and "WoodBlock" are blocks that protects the core block. The difference between them is the mass and the hp, "StoneBlock" are heavier and much stronger than "WoodBlock".

IV. Mechanism

A. How the car is moved

The game stores the car's moving data by momentum and angular momentum, and the game moves and rotates the car in every frame by checking the car's momentum and angular momentum. How much it moves can be calculated by the following equations:

$$\vec{x} = \frac{\vec{p}}{m} t_{frame} \tag{1}$$

$$\vec{\theta} = \frac{\vec{L}}{I} t_{frame} \tag{2}$$

Notice that every block in the game has mass. For core block it's 100kg, 50kg for stone and 10kg for wood. When rotation, it is rotated by the center of mass. This is how the center of mass is gotten:

$$\vec{x_{CM}} = \frac{\sum_{i=1}^{n} \vec{x_i} m_i}{\sum_{i=1}^{n} m_i} \tag{3}$$

In order to change the car's momentum and angular momentum, it needs to be applied force to it. How much the momentum and angular momentum is changed can be calculated by the following equations:

$$\Delta\vec{p} = \vec{F} t_{frame} \tag{4}$$

$$\Delta L = t_{frame}(\vec{r} \times \vec{F}) \tag{5}$$

While the momentum of inertia can be calculated by this equation:

$$I = \Sigma_{i=1}^{n} I_{CMi} + M_i D_i^2 \tag{6}$$

$$I_{CM} = \frac{1}{6} m a^2 \tag{7}$$

In the previous equation, m is the block's mass and a is the block's size. Because that every square is set at the size of 1, $a = 1$.

B. How the cars knows they collide each other

Because all shapes that build a car are square. We only need to check every block's points if one of the points is in the other car's body. If detected, we simply get the point in the other car and the previous point of the point which is inside the other car. Two points form a line segment, which represents how the car hits another car. By checking which line it collides to, we can get the direction of the force caused by the collision of both cars.The force caused by the collision can be calculated by the following equation:

Remember that when a collision fails, it's not a bug, it's a combo attack.

C. How the car is damaged

Every block has its own hp. 1000Hp for core block, 200 Hp for stone block, and 100 Hp for wood. When collision occurs, it will find which two blocks collide with each other. Then, it will calculate the damage the two blocks take. The damage can be calculated by the following equation:

$$Damage = (P_1 + P_2)/5000 \tag{8}$$

where

$$P_1 = \sqrt{P1_x^2 + P1_y^2} \tag{9}$$

$$P_2 = \sqrt{P2_x^2 + P2_y^2} \tag{10}$$

Notice that two blocks will take the same damage because of Newton's third law no matter how heavy or fast both cars go. Lastly, the same force will be applied to both cars from the colliding point. We assume that the force is applied in one frame of the game. The force can be calculated by the following equation:

$$\vec{F_{21}} = (-\frac{(1+e)[(\vec{v_2} - \vec{v_1}) \cdot \hat{n}]}{\frac{1}{m_1} + \frac{1}{m_2}} t_{frame})\hat{n} \tag{11}$$

The F is the force done to both cars participated in the collision, e is the coefficient of restitution, v is the velocity and m is the mass. n is the noraml vector which is perpendicular to the collision line pointing from car 1 to car 2.

D. Gravitational Force

At the middle of the battlefield, there is an object that has mass but with no volume, creating a gravitational force field. Two cars will be pulled into the center of the screen by applying force to them from the following equations:

$$\vec{F} = -\frac{GMm}{r^2}\hat{r} \tag{12}$$

F is the force applied to the car, G is the gravatitional constant, M is the mass of the center point, m is the car's mass, and r is the relative position pointing from the center of the gravity generating point to the car.

The gravitational force field can be adjusted by changing its mass.

E. Score calculation

When the battle ends, we can see the scoreboard at the center of the screen. The score of a player can be calculated by the following equations:

$$Score = HP_{remain}^{self} + HP_{loss}^{opponent} \tag{13}$$

Notice that the winner doesn't necessarily have a higher score, due to the way the formula above calculates the score.

ACKNOWLEDGMENT

Either sadly or happily, we are able to get nothing but fun from this game.

REFERENCES

[1] OpenAI. 2022. ChatGPT.