2D Shape Collision Simulation

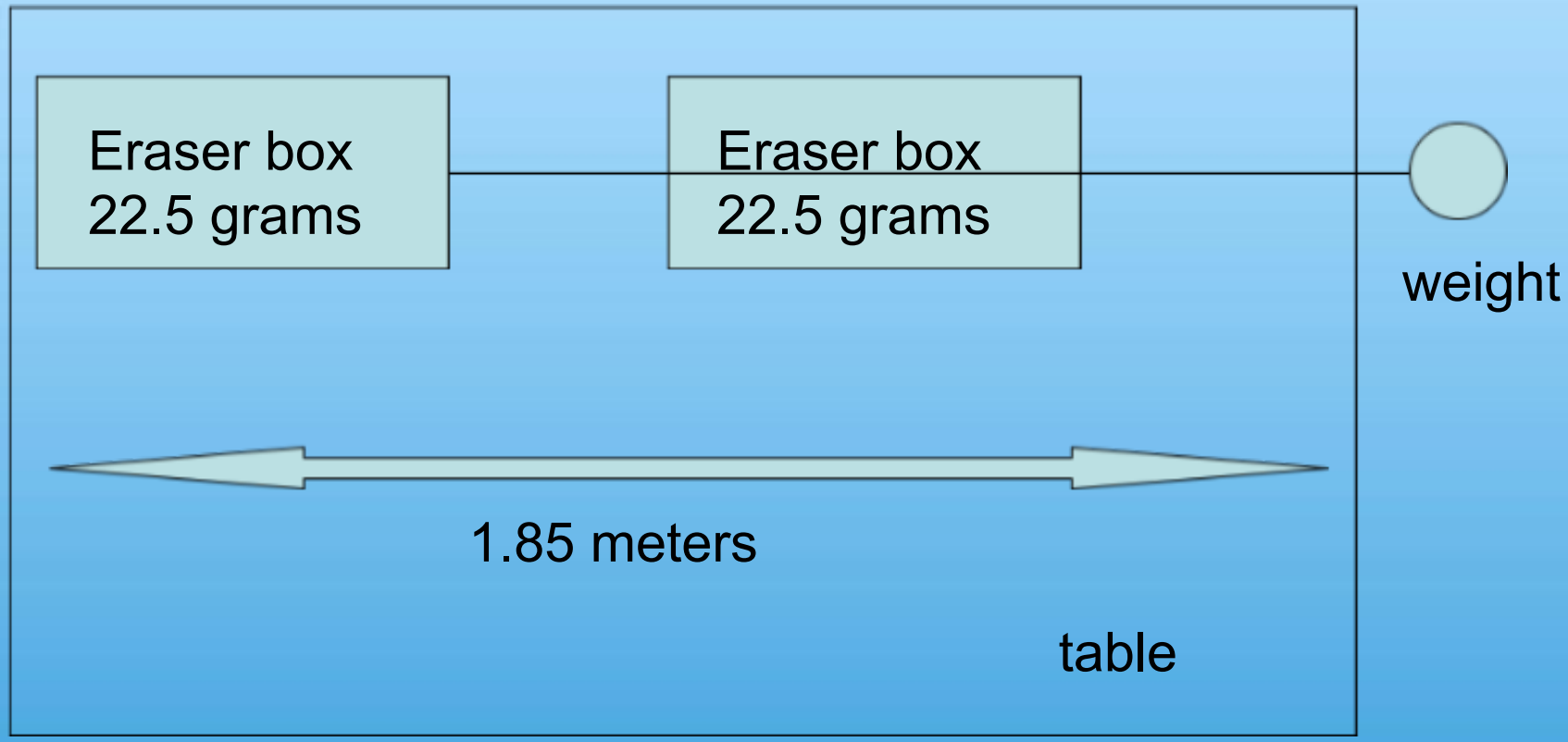
By: Ken Dopp, Tikhon Jelvis,
Gregory Nisbet, Jacob Taylor

Approach

Two aspects:

- Physical collision experiment with eraser boxes
  - provides real world data to corroborate simulation
- Interactive computer simulation
  - Simulates collisions with a variety of shapes

Experiment Setup



Experimental Variables

Independent:

- Initial distance and orientations

Dependent:

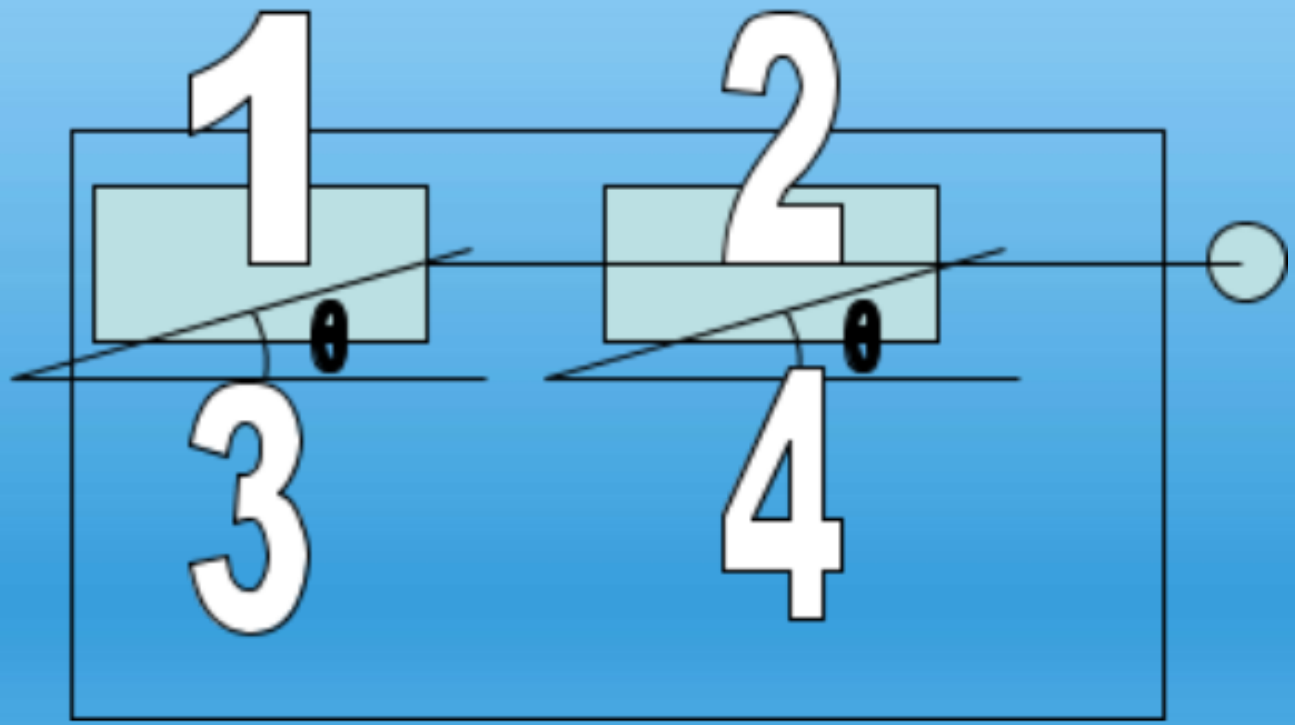
- Angles of objects
- Location of objects

Controlled:

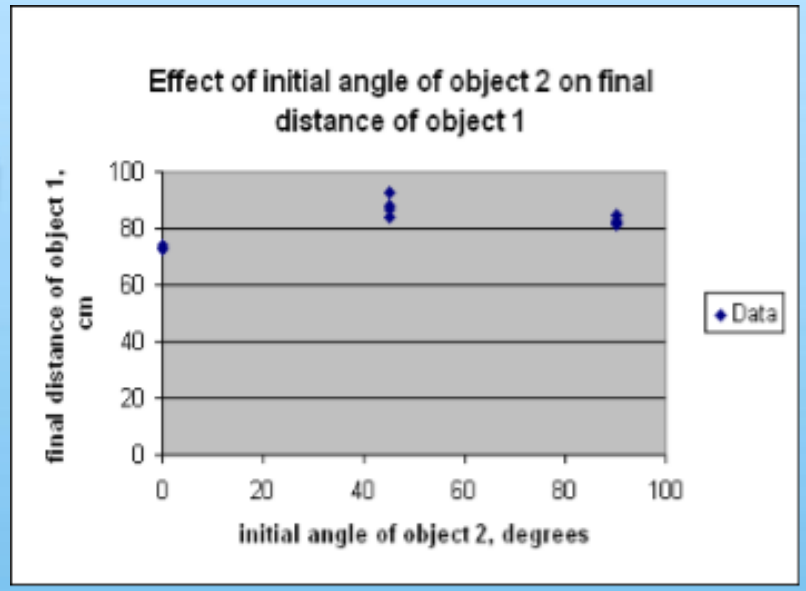
- Force applied to object
- Friction of table.

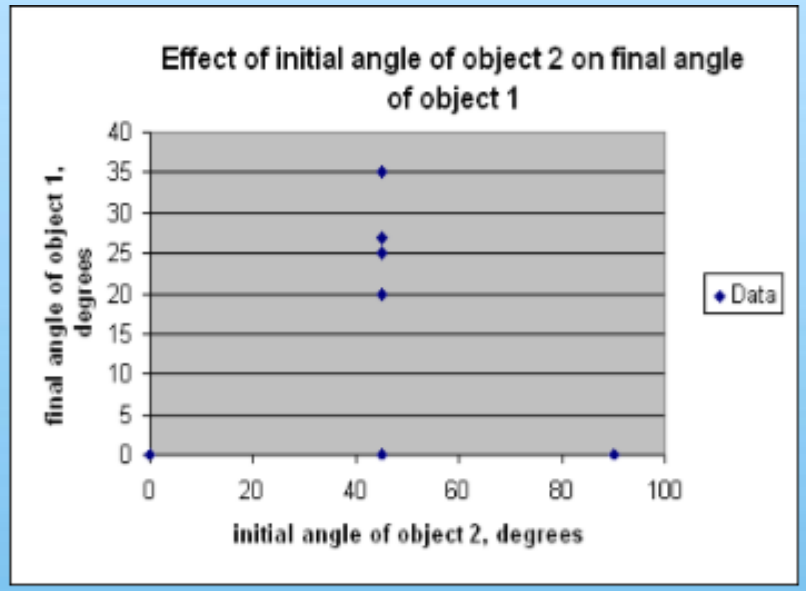
Measurements

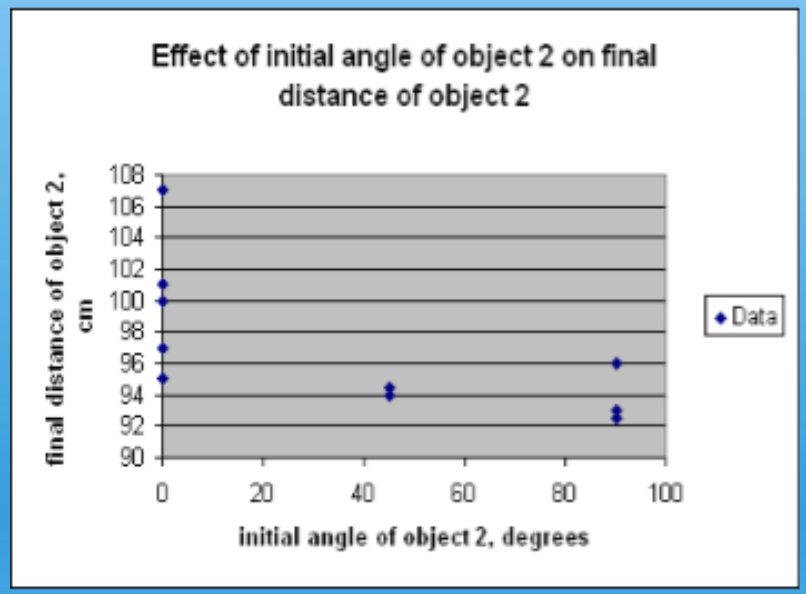
1. Distance of first box
2. Distance of second box
3. Angle of first box
4. Angle of second box

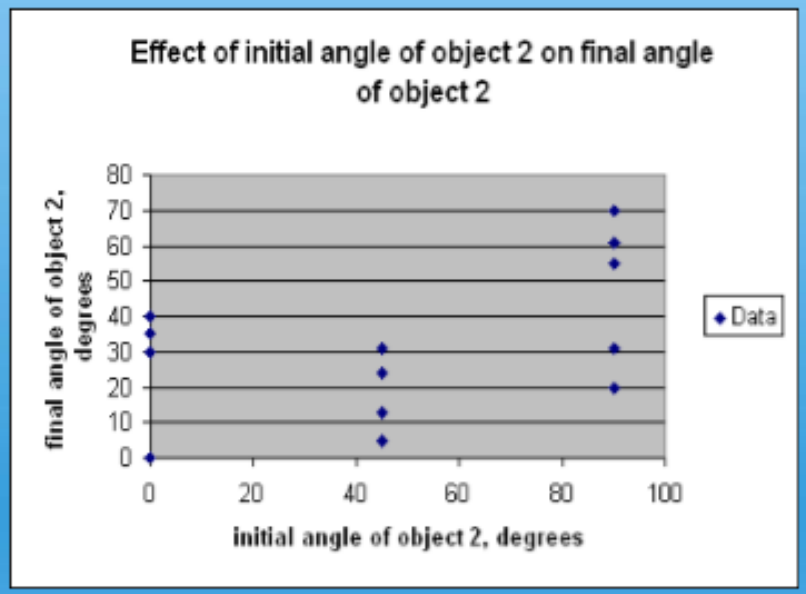


Data

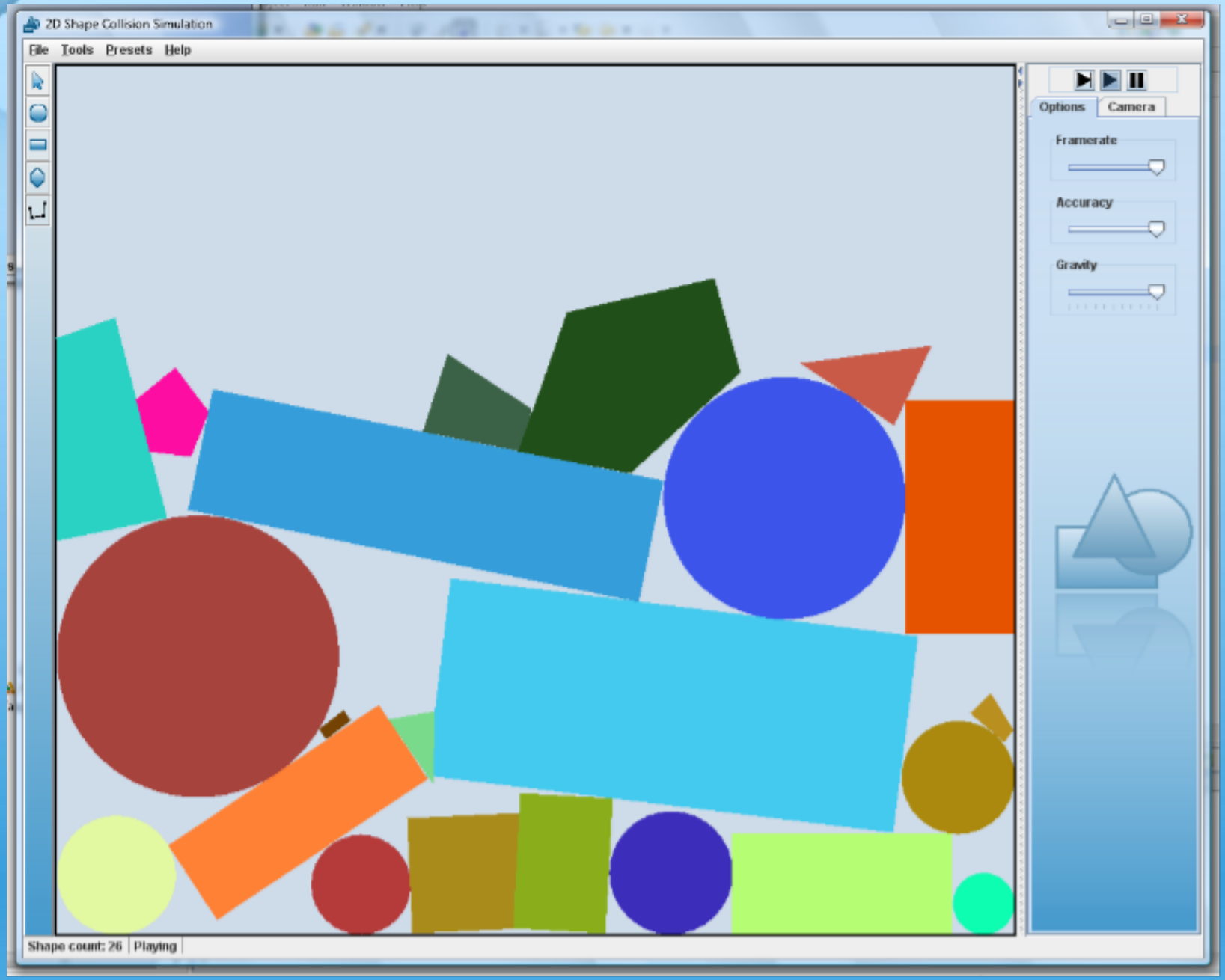

Effect of initial angle of object 2 on final distance of object 1
final distance of object 1, cm
initial angle of object 2, degrees
Data


Effect of initial angle of object 2 on final angle of object 1
final angle of object 1, degrees
initial angle of object 2, degrees
Data


Effect of initial angle of object 2 on final distance of object 2
final distance of object 2, cm
initial angle of object 2, degrees
Data


Effect of initial angle of object 2 on final angle of object 2
final angle of object 2, degrees
initial angle of object 2, degrees
Data

Simulation

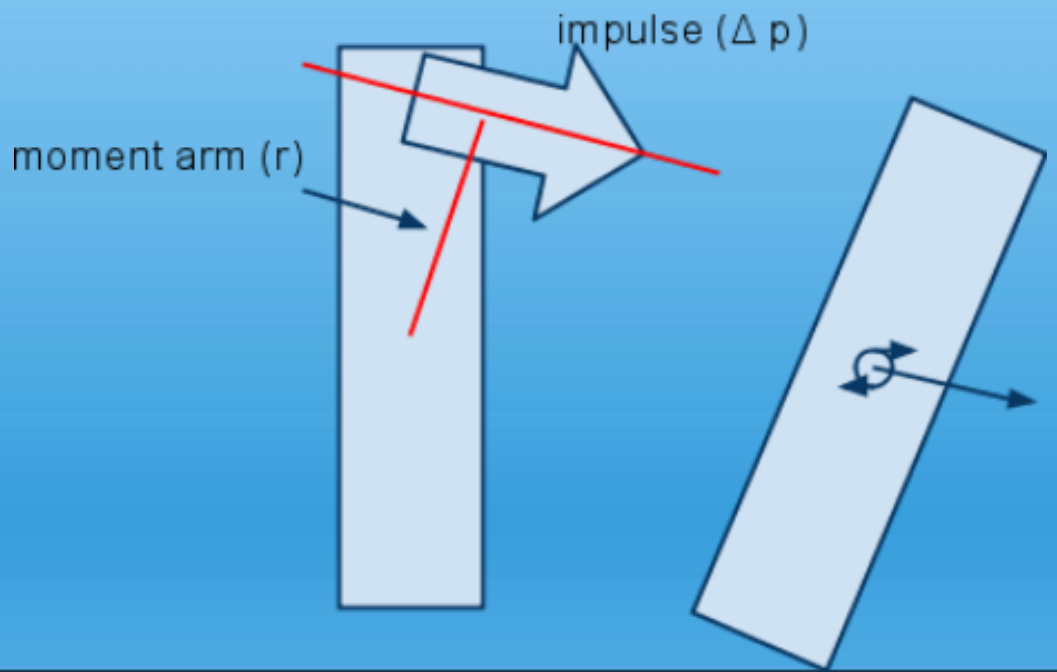

2D Shape Collision Simulation
File Tools Presets Help
Options Camera
Framerate
Accuracy
Gravity
Shape count: 26 Playing

Engine

- Contained in one package (separate from the GUI)
- Contains circles and polygons
- Each shape has a position, velocity, angular velocity, mass, and moment of inertia
  - Moment of inertia : measure of how difficult an object is to turn.

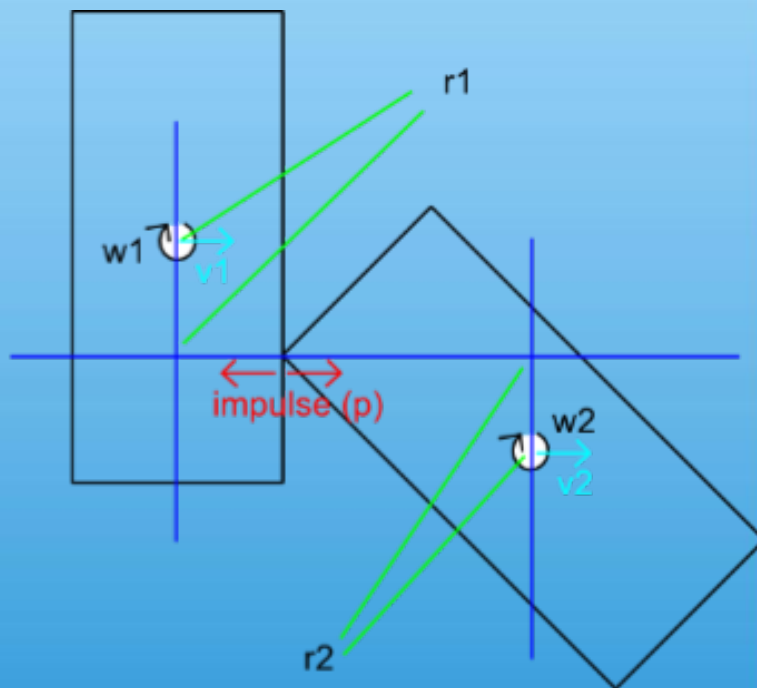
Impulses

- Impulse : a force applied instantaneously, measured in kg m/s
- Can change a shape's linear and/or angular velocity.
- Rotation depends on moment arm (r).
- $\Delta v = \Delta p/m$, $\Delta \omega = r\Delta p/I$ (I = moment of inertia)



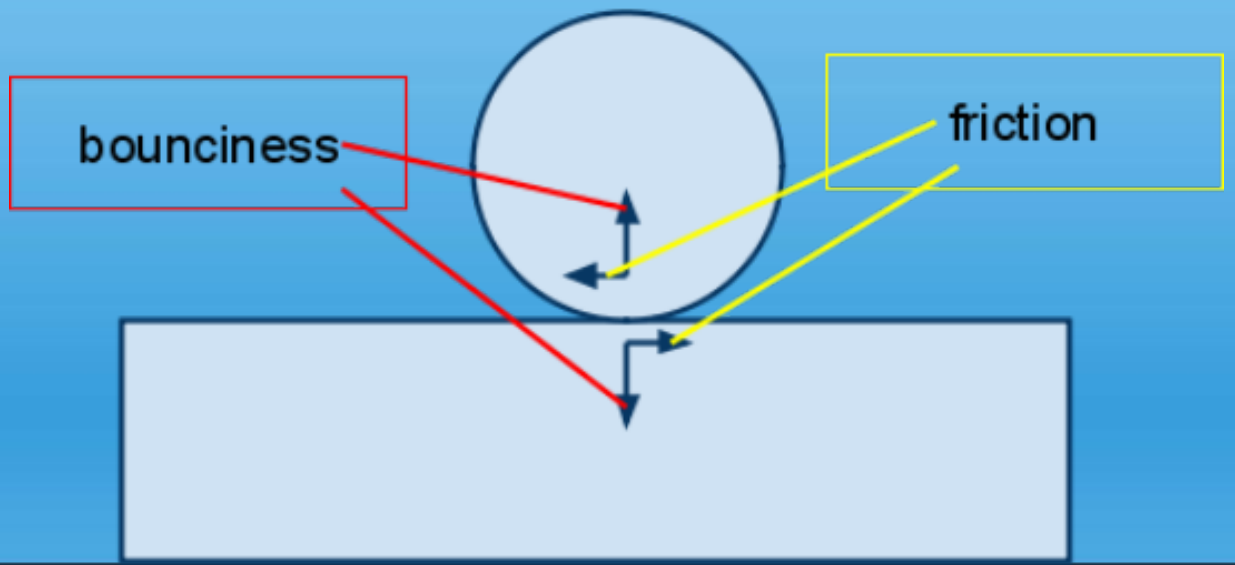
Collisions

- Impulses push the objects apart
- Elastic collisions preserve kinetic energy
- $E_k = \frac{1}{2}mv^2 + \frac{1}{2} I\omega^2$
- Both angular and linear velocity changes
- $$p = \frac{2(v_1 - v_2 + w_1 r_1 - w_2 r_2)}{(1/m_1 + 1/m_2 + r_{1\,2}/I_1 + r_{2\,2}/I_2)}$$

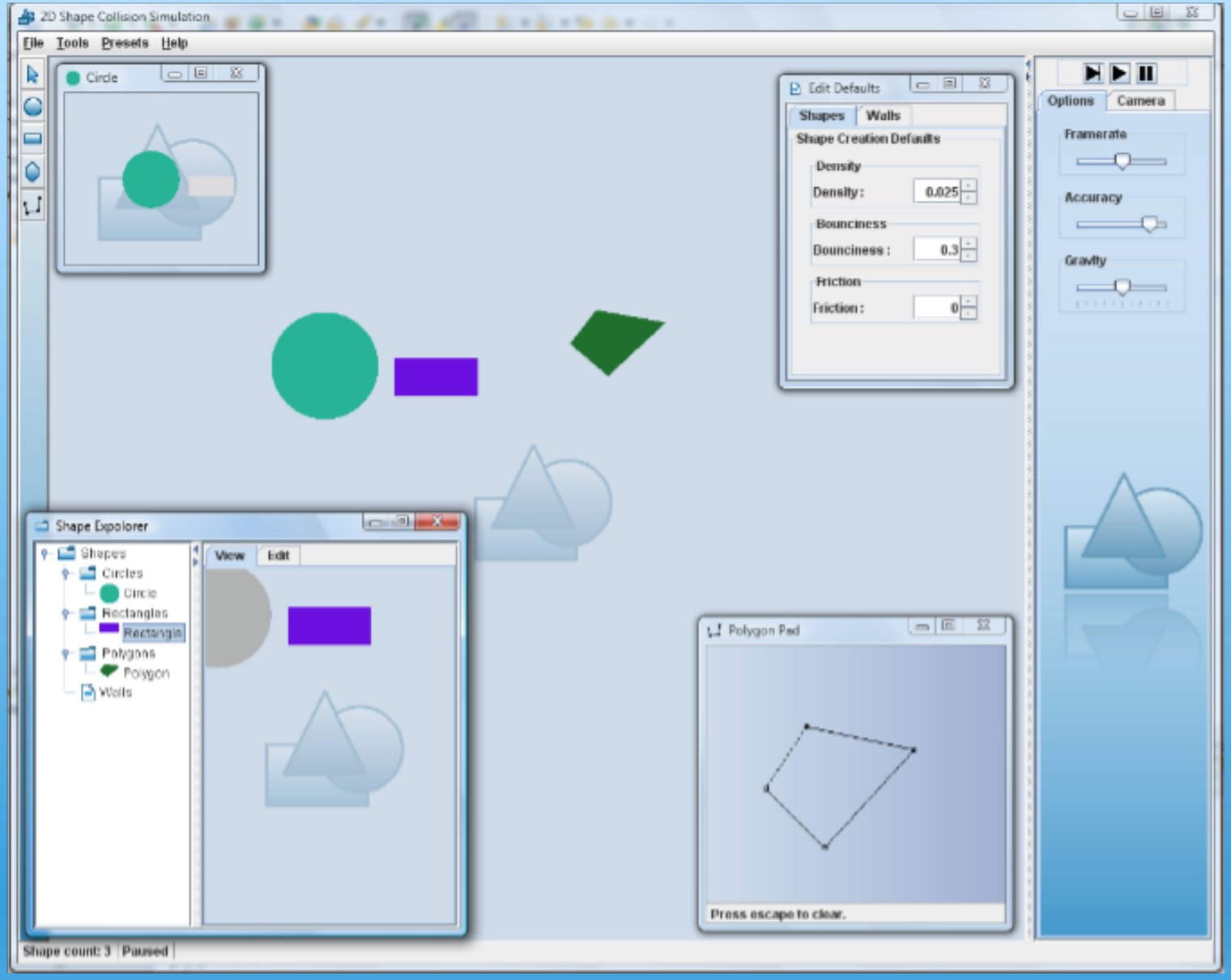


More on Collisions

- Objects have friction and bounciness
- $p = \frac{2(v1 - v2 + w1r1 - w2r2)}{(1/m1 + 1/m2 + r1\ 2\ /I1 + r2\ 2\ /I2)}$
- This formula is only for elastic collisions
- For inelastic collisions, the 2 is replaced by 1 + b1b2 (the bounciness factors, $0 \le b \le 1$)
- Friction: $\mu = f1f2$ (the friction factors, $0 \le f \le 1$)
- Friction impulse = μ * p (normal impulse)

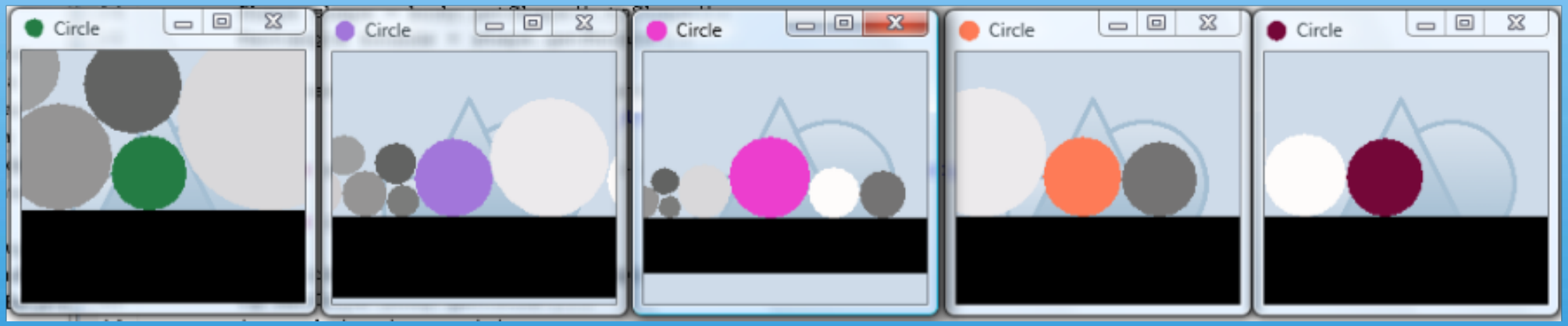


GUI


2D Shape Collision Simulation
File Tools Presets Help
Circle
Edit Defaults
Shapes Walls
Shape Creation Defaults
Density
Density: 0.025
Bounciness
Bounciness: 0.3
Friction
Friction: 0
Options Camera
Framerate
Accuracy
Gravity
Shape Explorer
Shapes
Circles
Circle
Rectangles
Rectangle
Polygons
Polygon
Walls
View Edit
Polygon Pad
Press escape to clear.
Shape count: 3 Paused

Structure

- Modular and extendable
- Model / view separation
  - GUI in different package from model
  - Model: the actual shapes and math
  - GUI: the user interface, uses the model



Synchronization

- Concurrency
  - Animation always requires multiple threads
- Manual synchronization
  - The two main lists needed to be synchronized manually

```
145    protected void step(double amount) {
146        // collide every body with every other body
147        // but don't do reverse collisions
148        // e.g. don't do both b.collide(a) and a.collide(b)
149        synchronized (bodies) {
150            for (int i = 0; i < bodies.size(); ++i) {
151                Body b1 = bodies.get(i);
152                for (int j = i + 1; j < bodies.size(); ++j) {
153                    Body b2 = bodies.get(j);
154                    b1.collide(b2);
155                }
156            }
157        }
158
159        synchronized (bodies) {
160            // step all bodies
161            for (Body b : bodies) {
162                b.step(amount);
163                // gravity
164                b.addImpulse(new Vector(0, 1 * amount * b.mass() * gravity));
165            }
166        }
167
168        synchronized (springs) {
169            // step all springs
170            for (Spring s : springs) {
171                s.step(amount);
172            }
173        }
174
175        synchronized (grabLock) {
176            if (grabbed != null) {
177                Vector mouse = new Vector(mouseX, mouseY);
178                Vector diff = mouse.subtract(grabbed.connectedPoint());
179                Vector result = diff.multiply(10 * amount);
180                grabbed.addImpulse(result);
181                grabbed
182                        .addImpulse(grabbed.getVelocity())
183                                .multiply(-10 * amount));
184            }
185        }
186    }
```

Documentation

- Javadocs (auto-generated html-based documentation for Java)
- Could help other people working on similar projects