5. Physics Libraries

박종화
suakii@gmail.com

Physics Libraries

- Learned about concepts from the world of physics — What is a vector? What is a force? What is a wave? etc.

- Understood the math and algorithms behind such concepts.

- Implemented the algorithms in Processing with an object-oriented approach.

Object

여러 운동들을 **시뮬레이션**

직접 설계한 월드의 **물리 법칙**을
창의적으로 정의

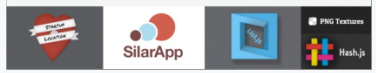
물리 라이브러리

Box2D

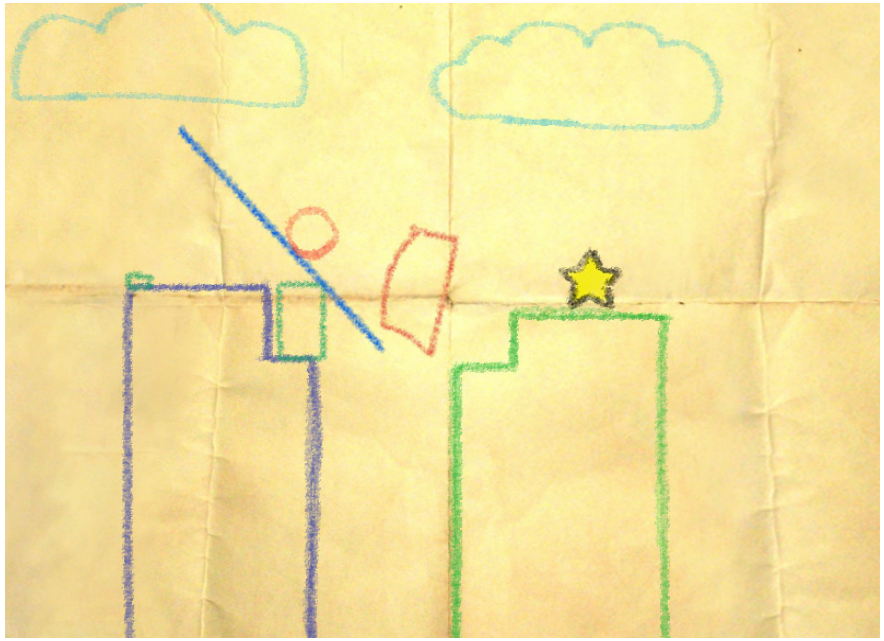
A Open Source Physics Engine

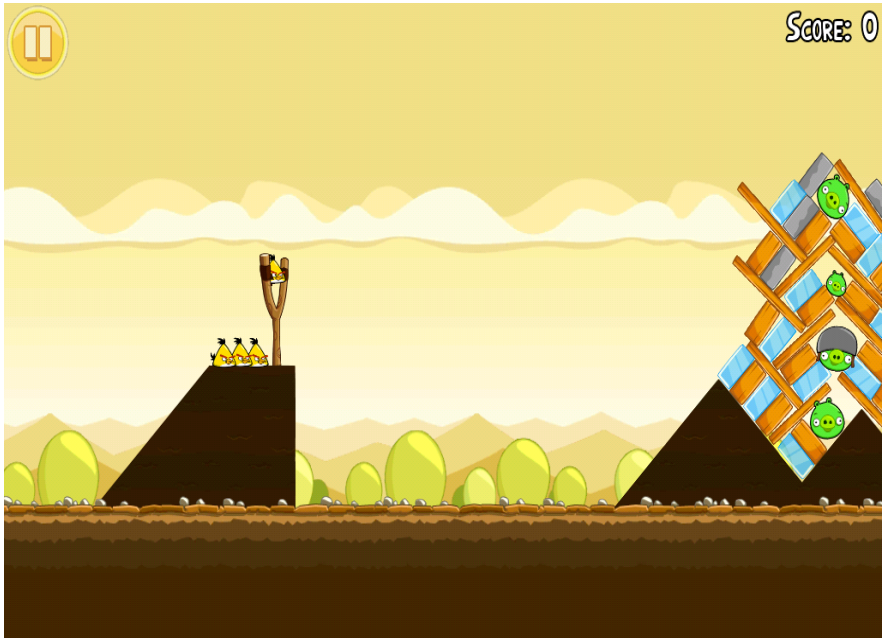
- www.box2d.org
- Erin Catto, 2007, C++
- zlib License (good open source stuff)
- Angry Birds, Tiny Wings etc.

Jonny Strömberg
jonnystromberg.com @javve



Box2D





시초: Erin Catto가 C++로 작성한 물리 튜토리얼
'진짜' 물리엔진
그래픽, 픽셀 X
Input: 수 Output: 수


Highscore: 73190
Score: 0
ANGRY BIRDS
ROVIO
TINY WINGS
LOADING...
NEW GAME
CONTINUE
SELECT LEVEL
LEVEL EDITOR
OPTION
HOW TO PLAY
CRAYON PHYSICS Deluxe
TO WEB
©Kloonigames
Licensed exclusively to Hudson Soft for Apple iPhone and iPod touch / ©HUDSON SOFT
TNT
Jonny Strömberg
jonnystromberg.com @javve
SilarApp
PNG Textures
Hash.js

What Is Box2D?

- Box2D is open source C++ engine for simulating rigid bodies in 2D.
- Box2D is developed by Erin Catto and has the zlib license. While the zlib license does not require acknowledgement, we encourage you to give credit to Box2D in your product.

Good thing to know

Otherwise stuff will be messy

- No pixels. Only meters.
- Convex, not conaves. 
- Moving objects 0.1-10 m. Static 0.1-50 m.
- Only physics, no graphics.

Box2D

- www.box2d.org

- Box2D의 단위 체계
  - Meters, kilograms, seconds, etc.

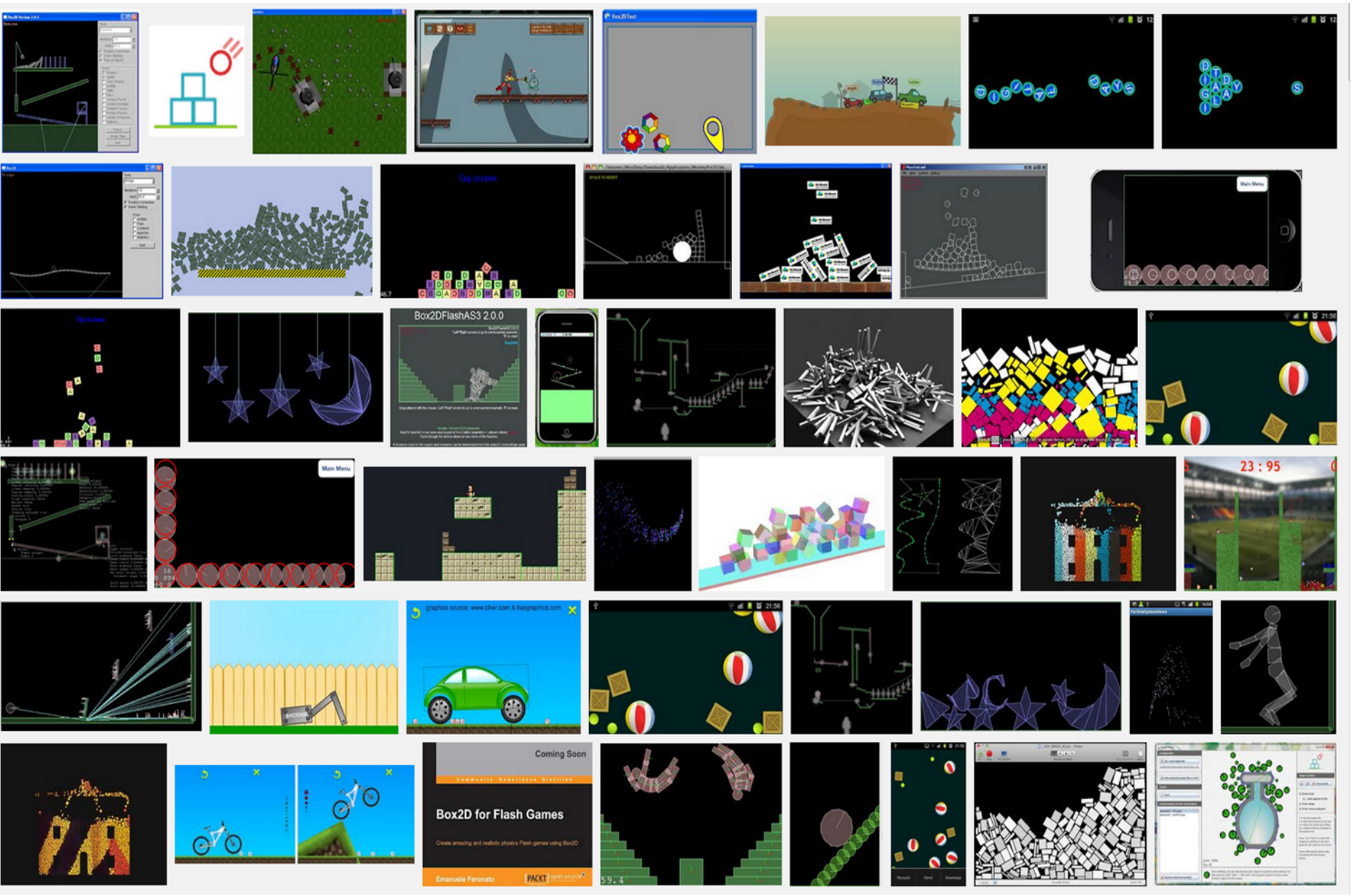
"Bodies are soft bags with hard fixtures inside."

Bodies

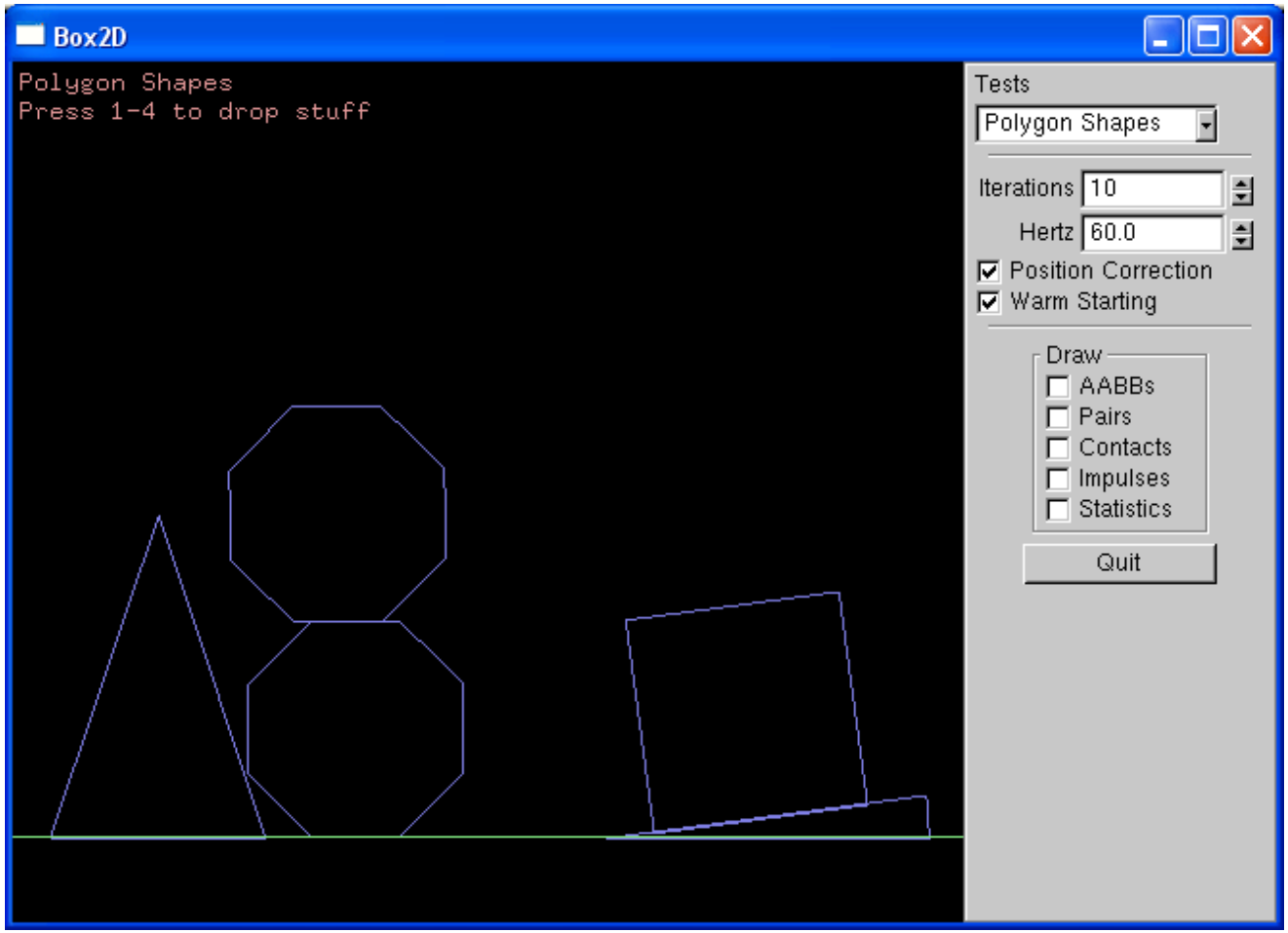
- Contains fixture(s)
- Position
- Angle
- Static or dynamic

Fixtures

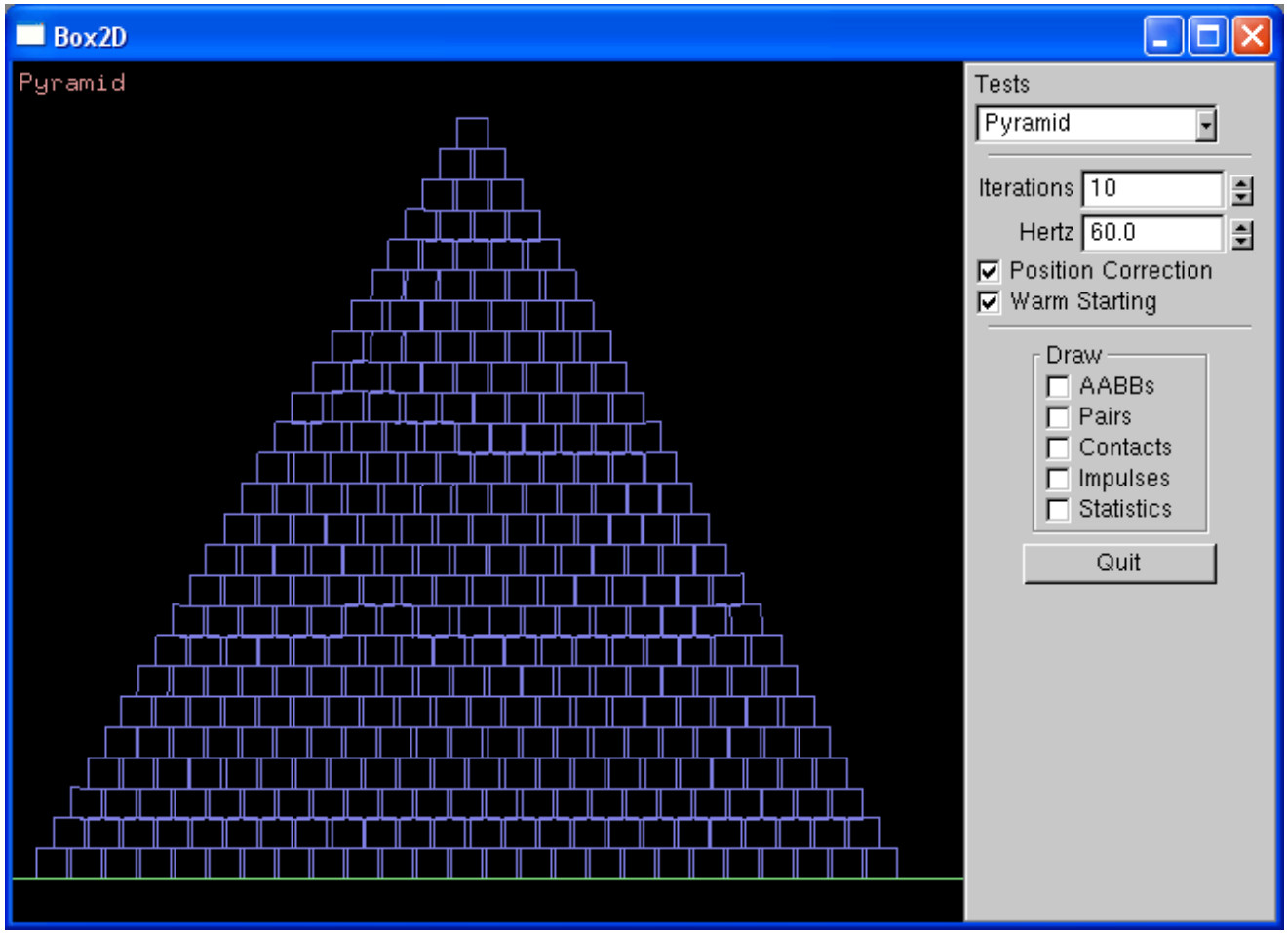
- Density
- Friction
- Has a shape
- Collision


Box2DFlashAS3 2.0.0
Main Menu
Coming Soon
Box2D for Flash Games
Create amazing and realistic physics Flash games using Box2D
Emanuele Feronato
PACKT

Box2D


Box2D
Polygon Shapes
Press 1-4 to drop stuff
Tests
Polygon Shapes
Iterations 10
Hertz 60.0
Position Correction
Warm Starting
Draw
AABBs
Pairs
Contacts
Impulses
Statistics
Quit

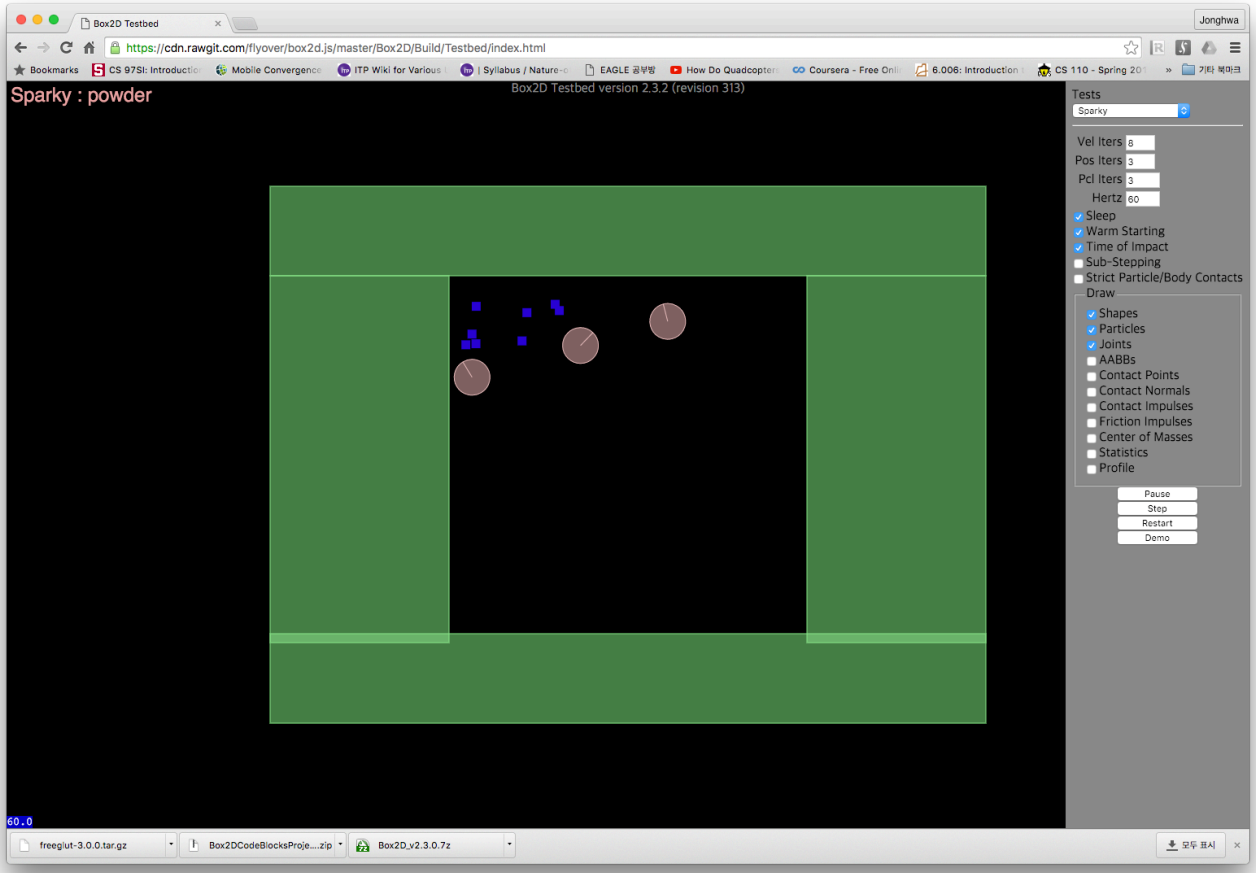
Box2D


Box2D
Pyramid
Tests
Pyramid
Iterations 10
Hertz 60.0
Position Correction
Warm Starting
Draw
AABBs
Pairs
Contacts
Impulses
Statistics
Quit

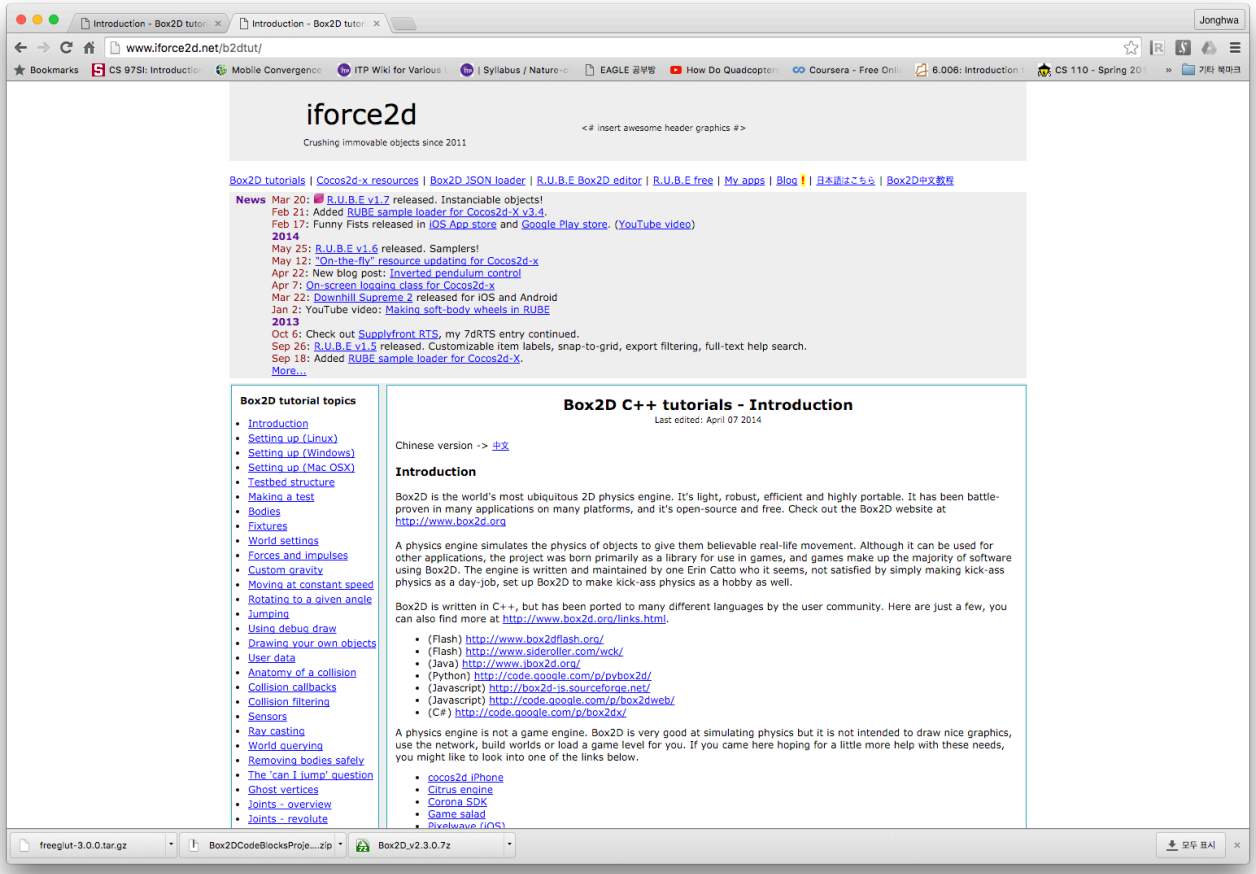
Box2D Testbed

- https://cdn.rawgit.com/flyover/box2d.js/master/Box2D/Build/Testbed/index.html

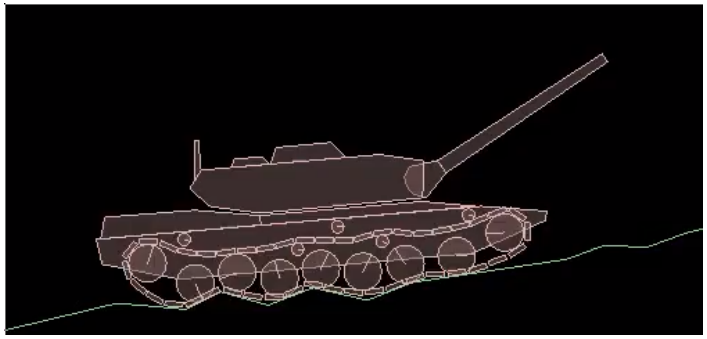
Box2d.js


Box2D Testbed
Jonghwa
https://cdn.rawgit.com/flyover/box2d.js/master/Box2D/Build/Testbed/index.html
Sparky : powder
Box2D Testbed version 2.3.2 (revision 313)
Tests
Sparky
Vel Iters 8
Pos Iters 3
Pcl Iters 3
Hertz 60
Sleep
Warm Starting
Time of Impact
Sub-Stepping
Strict Particle/Body Contacts
Draw
Shapes
Particles
Joints
AABBs
Contact Points
Contact Normals
Contact Impulses
Friction Impulses
Center of Masses
Statistics
Profile
Pause
Step
Restart
Demo
60.0
freeglut-3.0.0.tar.gz
Box2DCodeBlocksProje....zip
Box2D_v2.3.0.7z

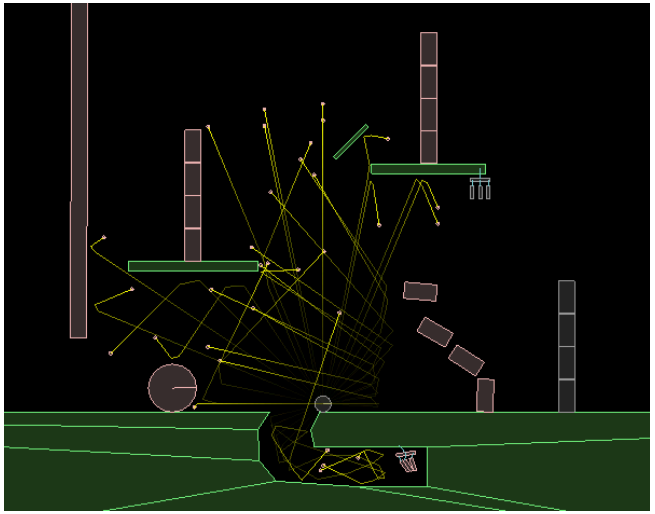
iforce2d


Introduction - Box2D tutor
Introduction - Box2D tutor
Jonghwa
www.iforce2d.net/b2dtut/
Bookmarks
iforce2d
Crushing immovable objects since 2011
<# insert awesome header graphics #>
Box2D tutorials | Cocos2d-x resources | Box2D JSON loader | R.U.B.E Box2D editor | R.U.B.E free | My apps | Blog | 日本語はこちら | Box2D中文教程
News
Mar 20: R.U.B.E v1.7 released. Instanciable objects!
Feb 21: Added RUBE sample loader for Cocos2d-X v3.4.
Feb 17: Funny Fists released in iOS App store and Google Play store. (YouTube video)
2014
May 25: R.U.B.E v1.6 released. Samplers!
May 12: "On-the-fly" resource updating for Cocos2d-x
Apr 22: New blog post: Inverted pendulum control
Apr 7: On-screen logging class for Cocos2d-x
Mar 22: Downhill Supreme 2 released for iOS and Android
Jan 2: YouTube video: Making soft-body wheels in RUBE
2013
Oct 6: Check out Supplyfront RTS, my 7dRTS entry continued.
Sep 26: R.U.B.E v1.5 released. Customizable item labels, snap-to-grid, export filtering, full-text help search.
Sep 18: Added RUBE sample loader for Cocos2d-X.
More...
Box2D tutorial topics
Introduction
Setting up (Linux)
Setting up (Windows)
Setting up (Mac OSX)
Testbed structure
Making a test
Bodies
Fixtures
World settings
Forces and impulses
Custom gravity
Moving at constant speed
Rotating to a given angle
Jumping
Using debug draw
Drawing your own objects
User data
Anatomy of a collision
Collision callbacks
Collision filtering
Sensors
Ray casting
World querying
Removing bodies safely
The 'can I jump' question
Ghost vertices
Joints - overview
Joints - revolute
Box2D C++ tutorials - Introduction
Last edited: April 07 2014
Chinese version -> 中文
Introduction
Box2D is the world's most ubiquitous 2D physics engine. It's light, robust, efficient and highly portable. It has been battle-proven in many applications on many platforms, and it's open-source and free. Check out the Box2D website at http://www.box2d.org
A physics engine simulates the physics of objects to give them believable real-life movement. Although it can be used for other applications, the project was born primarily as a library for use in games, and games make up the majority of software using Box2D. The engine is written and maintained by one Erin Catto who it seems, not satisfied by simply making kick-ass physics as a day-job, set up Box2D to make kick-ass physics as a hobby as well.
Box2D is written in C++, but has been ported to many different languages by the user community. Here are just a few, you can also find more at http://www.box2d.org/links.html.
(Flash) http://www.box2dflash.org/
(Flash) http://www.sideroller.com/wck/
(Java) http://www.jbox2d.org/
(Python) http://code.google.com/p/pybox2d/
(Javascript) http://box2d-js.sourceforge.net/
(Javascript) http://code.google.com/p/box2dweb/
(C#) http://code.google.com/p/box2dx/
A physics engine is not a game engine. Box2D is very good at simulating physics but it is not intended to draw nice graphics, use the network, build worlds or load a game level for you. If you came here hoping for a little more help with these needs, you might like to look into one of the links below.
cocos2d iPhone
Citrus engine
Corona SDK
Game salad
freeglut-3.0.0.tar.gz
Box2DCodeBlocksProje....zip
Box2D_v2.3.0.7z
모두 표시

Iforce2d Advanced Topics



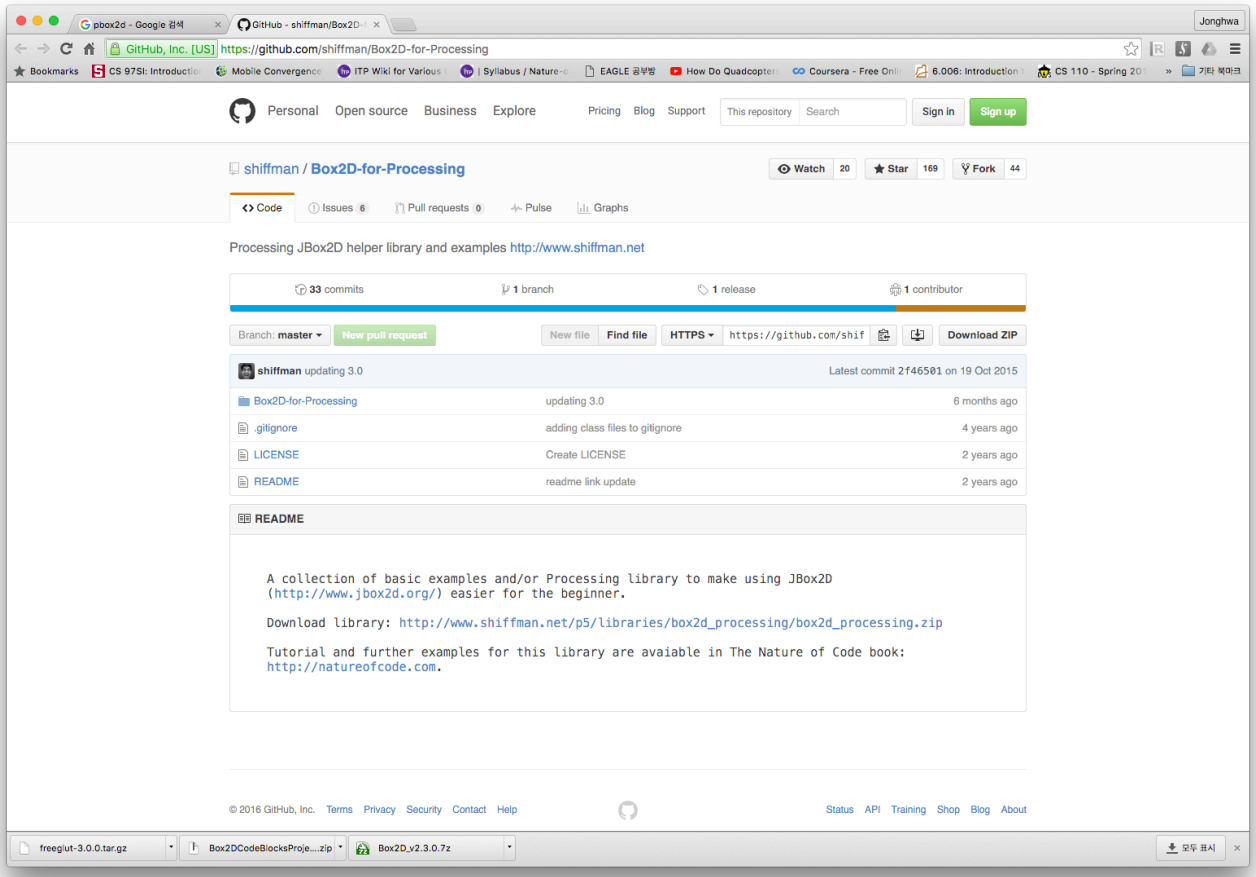
https://www.youtube.com/watch?v=0awh5MFJwQo#t=2m12s



Box2D

- Erin: “Hi Peter, could you tell me which physics engine Angy Birds uses?”
- Peter: “Box2D”
- Erin: “Great. Would you consider giving credit to Box2D in your game?”
- Peter: “Yes, of course”
- Erin: “Thank you! By the way, I am Erin Catto the creator of Box2D”
- Peter: “Great! I would like to talk to you after the session”

PBox2D


pbox2d - Google 검색
GitHub - shiffman/Box2D-for-Processing
Jonghwa
GitHub, Inc. [US] https://github.com/shiffman/Box2D-for-Processing
Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up
shiffman / Box2D-for-Processing
Watch 20 Star 169 Fork 44
Code Issues 6 Pull requests 0 Pulse Graphs
Processing JBox2D helper library and examples http://www.shiffman.net
33 commits 1 branch 1 release 1 contributor
Branch: master New pull request New file Find file HTTPS https://github.com/shif Download ZIP
shiffman updating 3.0 Latest commit 2f46501 on 19 Oct 2015
Box2D-for-Processing updating 3.0 6 months ago
.gitignore adding class files to gitignore 4 years ago
LICENSE Create LICENSE 2 years ago
README readme link update 2 years ago
README
A collection of basic examples and/or Processing library to make using JBox2D
(http://www.jbox2d.org/) easier for the beginner.
Download library: http://www.shiffman.net/p5/libraries/box2d_processing/box2d_processing.zip
Tutorial and further examples for this library are avaiable in The Nature of Code book:
http://natureofcode.com.
© 2016 GitHub, Inc. Terms Privacy Security Contact Help
Status API Training Shop Blog About
freeglut-3.0.0.tar.gz
Box2DCodeBlocksProje....zip
Box2D_v2.3.0.7z
모두 표시

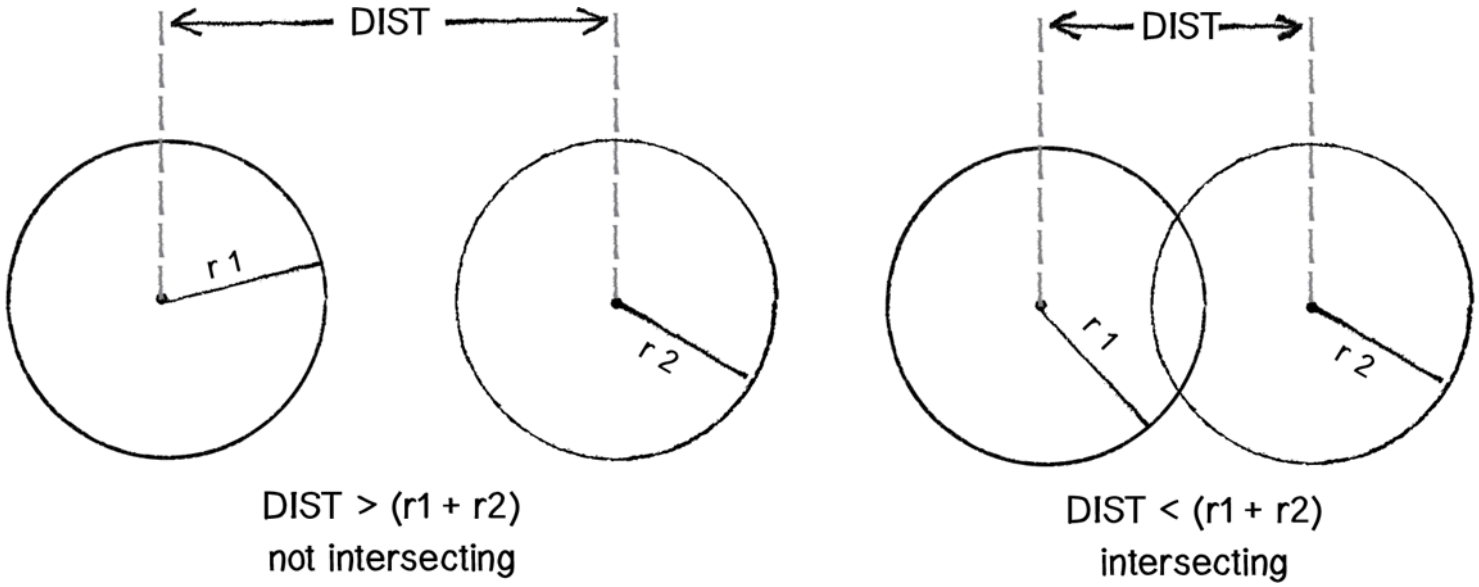
PBox2D

- JBox2D: Java Wrapper
  - 프로세싱에서 바로 사용이 가능

- PBox2D : Processing Wrapper
  - JBox2D를 보다 쉽게 사용
  - A Layer library between JBox2D and Processing
  - PBox2D is not a Processing wrapper for all of Box2D

Why Box2D?

- 충돌을 다룰 수 있다.



Box2D Basics

- Setup
  - 객체 생성

- Draw
  - 힘을 계산
  - 객체들에 힘을 적용
  - 가속도와 속도를 기반으로 객체 위치 변경
  - 객체를 그린다.

Box2D Basics

- Setup
  - 객체를 생성
- Draw
  - 객체를 그린다.

Box2D

- Setup
  - 픽셀 단위로 모든 객체를 만들어 준다.
  - 픽셀 단위를 Box2D 세계의 단위로 변환
- Draw
  - 객체의 위치를 Box2D에게 물어봄
  - Box2D의 단위를 픽셀로 변환
  - 객체를 그린다.

Box2D


www.Bandicam.co.kr
SCORE: 0
1. World
4. Fixture
2. Body
5. Joint
3. Shape

Box2D

- World:
  - 물리 시뮬레이션을 관리, 좌표공간을 관리하고 생성한 모든 객체를 저장(전체 공간)
- Body:
  - Body는 좌표와 속도를 가진다. (추상적인 하나의 객체)
- Shape:
  - 보디에 붙어서 충돌과 관련된 지오메트리 처리
  - 하나가 될 수 도 있고 여러 개가 될 수 도 있음(사람 – 팔, 다리)
- Fixture:
  - 셰이프를 보디에 붙이고 밀도, 마찰, 복원력 등의 요소를 적용
  - Body와 Shape를 연결해주는 본드?
- Joint:
  - 2개의 보디(또는 1개의 보디와 월드)를 연결해서 상호 작용을 일으키게 함.
- Vec2:
  - Box2D에서 사용할 수 있는 벡터

Pvector VS Vec2

PVector	Vec2

`PVector a = new PVector(1,-1);`
`PVector b = new PVector(3,4);`
`a.add(b);`
	`Vec2 a = new Vec2(1,-1);`
	`Vec2 b = new Vec2(3,4);`
	`a.addLocal(b);`

`PVector a = new PVector(1,-1);`
`PVector b = new PVector(3,4);`
`PVector c = PVector.add(a,b);`
	`Vec2 a = new Vec2(1,-1);`
	`Vec2 b = new Vec2(3,4);`
	`Vec2 c = a.add(b);`

PVector	Vec2

`PVector a = new PVector(1,-1);`
`float n = 5;`
`a.mult(n);`
	`Vec2 a = new Vec2(1,-1);`
	`float n = 5;`
	`a.mulLocal(n);`

`PVector a = new PVector(1,-1);`
`float n = 5;`
`PVector c = PVector.mult(a,n);`
	`Vec2 a = new Vec2(1,-1);`
	`float n = 5;`
	`Vec2 c = a.mul(n);`

PVector	Vec2

`PVector a = new PVector(1,-1);`
`float m = a.mag();`
`a.normalize();`
	`Vec2 a = new Vec2(1,-1);`
	`float m = a.length();`
	`a.normalize();`

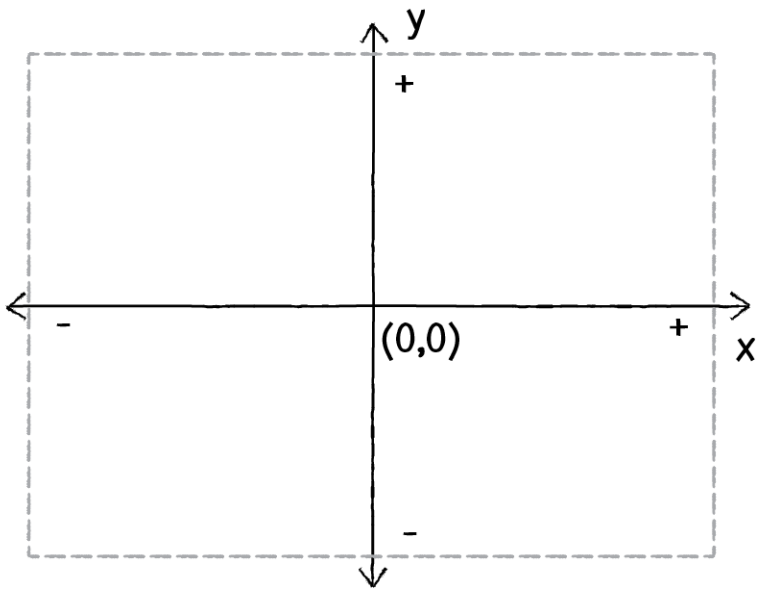
Living in a Box2D World

- World
  - 모든 것을 관리하는 객체
  - 모든 객체의 좌표를 관리
  - 시간 체크

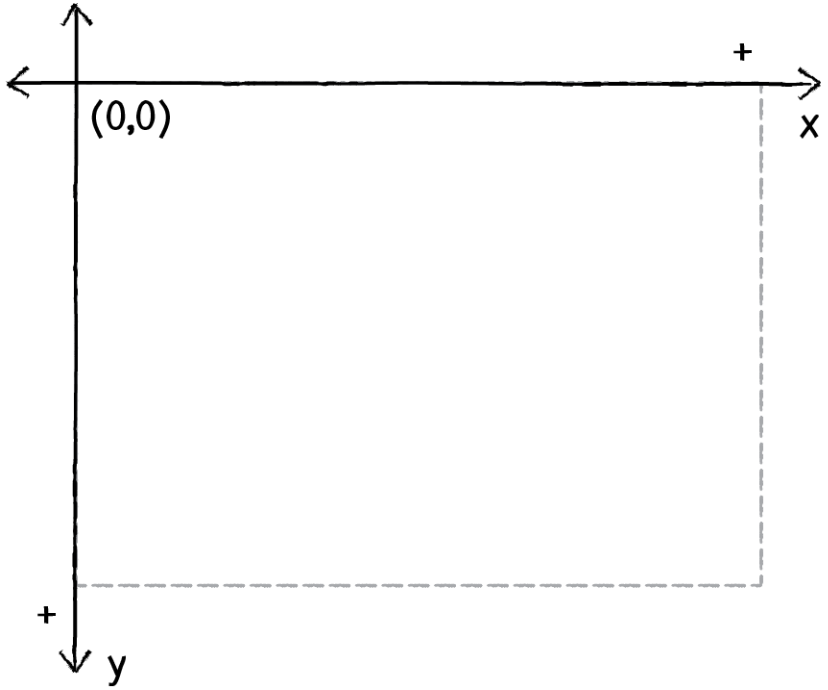
World

```
import shiffman.box2d.*;
import org.jbox2d.collision.shapes.*;
import org.jbox2d.common.*;
import org.jbox2d.dynamics.*;

Box2DProcessing box2d;

void setup() {
        box2d = new Box2DProcessing(this);
        box2d.createWorld();
}
```

World



Box 2D world



Processing World

Coordinate Convert

Task	Function
Convert location from World to Pixels	Vec2 coordWorldToPixels(Vec2 world)
Convert location from World to Pixels	Vec2 coordWorldToPixels(float worldX, float worldY)
Convert location from Pixels to World	Vec2 coordPixelsToWorld(Vec2 screen)
Convert location from Pixels to World	Vec2 coordPixelsToWorld(float pixelX, float pixelY)
Scale a dimension (such as height, width, or radius) from Pixels to World	float scalarPixelsToWorld(float val)
Scale a dimension from World to Pixels	float scalarWorldToPixels(float val)

Building a Box2D Body

- Mover 객체처럼 공간에 자신의 위치를 가지고 힘을 받게 된다.
- It's important to note, however, that a body has no geometry; it isn't anything physical.
  - 물리적인 반응을 하지 않는다.
- Shape를 붙여서 물리적인 반응 가능

Building a Box2D Body

- Step 1: Define a body
  - BodyDef 객체를 생성
  - 바디의 속성을 정의한다.

- Make a body definition before making a Body
  - BodyDef bd = new BodyDef();

Building a Box2D Body

- Step 2: Configure the body definition
  - 바디의 특성 또는 속성을 정의

```
Vec2 center = box2d.coordPixelsToWorld(width/2,height/2));

bd.position.set(center);
```

Building a Box2D Body

- Body's type
  - Dynamic
    - 모든 물리 시뮬레이션이 적용되며, 제일 많이 사용된다. 동적 보디는 월드를 움직이며 다른 보디와 상호 작용 가능. 외부 힘에도 반응
  - Static
    - 움직이지 않음. 틀과 같은 환경을 만들 때.
  - Kinematic
    - 속도를 직접 설정해서 수동으로 움직일 수 있다. 사용자가 움직이게 만들 수 있는 요소를 월드 안에 넣고 싶을 때 사용. 동적 보디와만 충돌을 일으킴

Building a Box2D Body

- bd.fixedRotation = true;
- bd.linearDamping = 0.8; //선형 감쇠
- bd.angularDamping = 0.9; // 회전 감소
- bd.bullet = true;

Building a Box2D Body

- Step 3: Create Body
  - Body body = box2d.createBody(bd);
- Step 4: set other..
  - body.setLinearVelocity(new Vec2(0, 3));
  - body.setAngularVelocity(1.2);

Bodies & Shapes & Fixtures

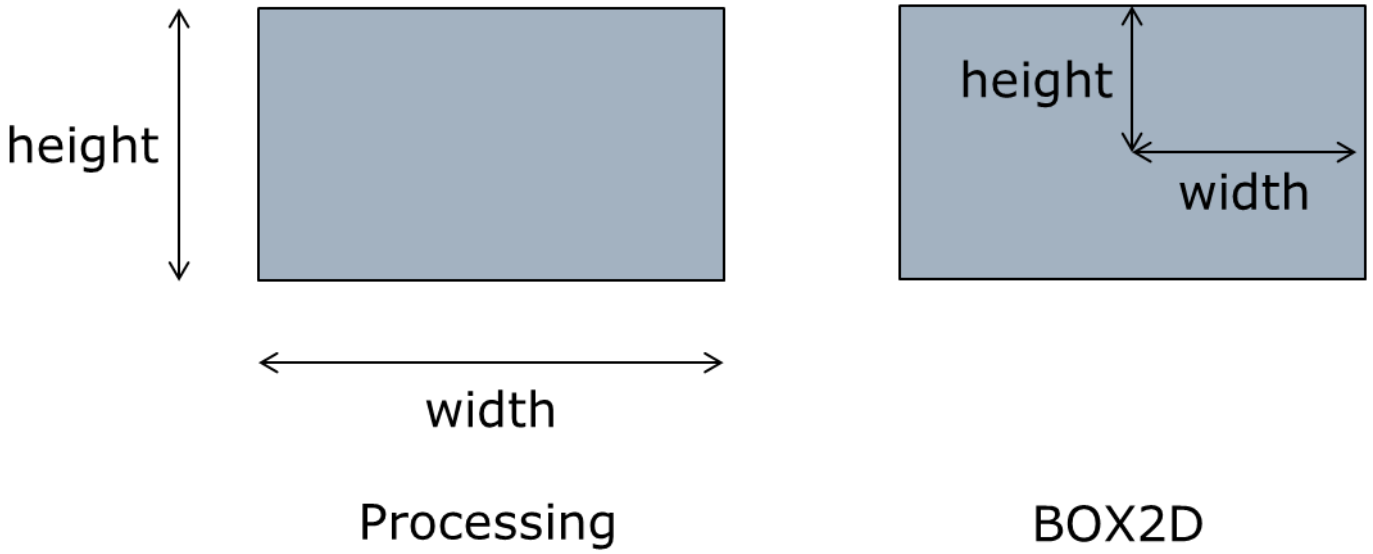
- 보디 자체는 물리적인 반응을 하지 못한다.
  - ㅠㅠ
- 보디에 질량을 부여하려면 셰이프를 만들고 픽스처로 셰이프를 보디에 부착한다.

Bodies & Shapes & Fixtures

- Step 1 : Define a shape
  - PolygonShape ps = new PolygonShape();

```
float box2Dw = box2d.scalarPixelsToWorld(150);
float box2Dh = box2d.scalarPixelsToWorld(100);


ps.setAsBox(box2Dw, box2Dh);
```

Width Height



Bodies & Shapes & Fixtures

- Step 2: Create a fixture
  - 생성한 셰이프를 보디에 붙이려면 픽스처를 사용
  - FixtureDef
    - FixtureDef fd = new FixtureDef();
    - fd.shape = ps;
      - The fixture is assigned the PolygonShape we just made

```
fd.friction = 0.3;
```
마찰계수

```
fd.restitution = 0.5;
```
탄성계수

```
fd.density = 1.0;
```
밀도

Bodies & Shapes & Fixtures

- Step 3: Attach the shape to the body with the fixture
  - body.createFixture(fd);

Summary

- Define a body using a BodyDef object (set any properties, such as location).
- Create the Body object from the body definition.
- Define a Shape object using PolygonShape, CircleShape, or any other shape class.
- Define a fixture using FixtureDef and assign the fixture a shape (set any properties, such as friction, density, and restitution).
- Attach the shape to the body.

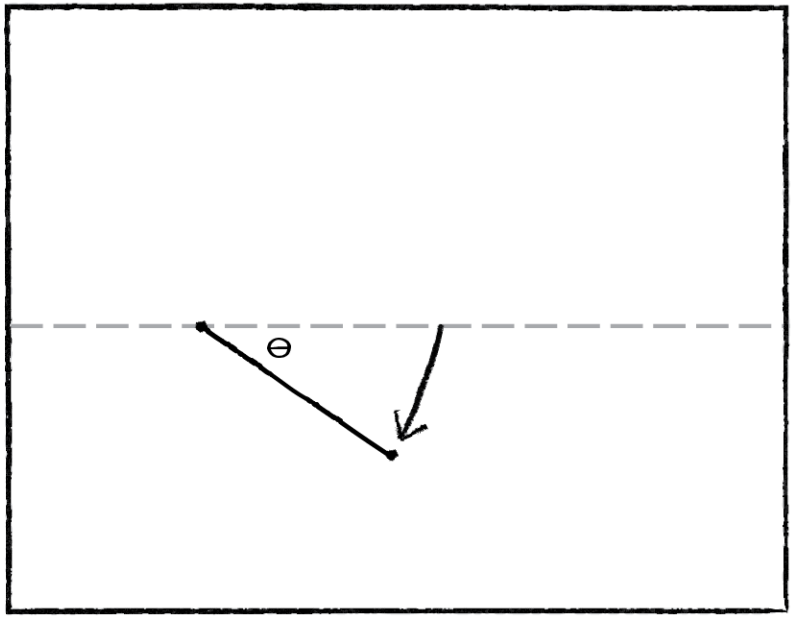
Box2D and Processing

- Box2D는 물리와 관련된 연산만 책임
- 결국 화면에 그리는 것은 우리의 책임

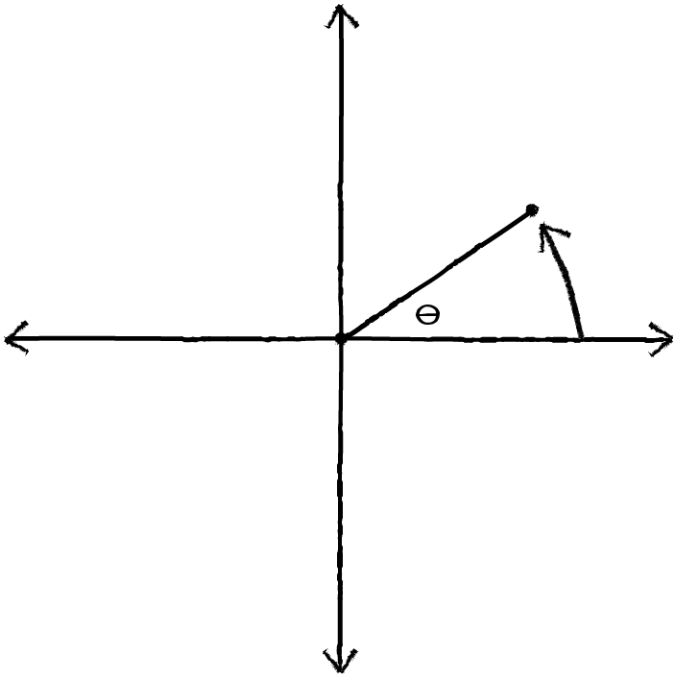
예제

- box2d_exercise
- box2d_exercise_solved

Example

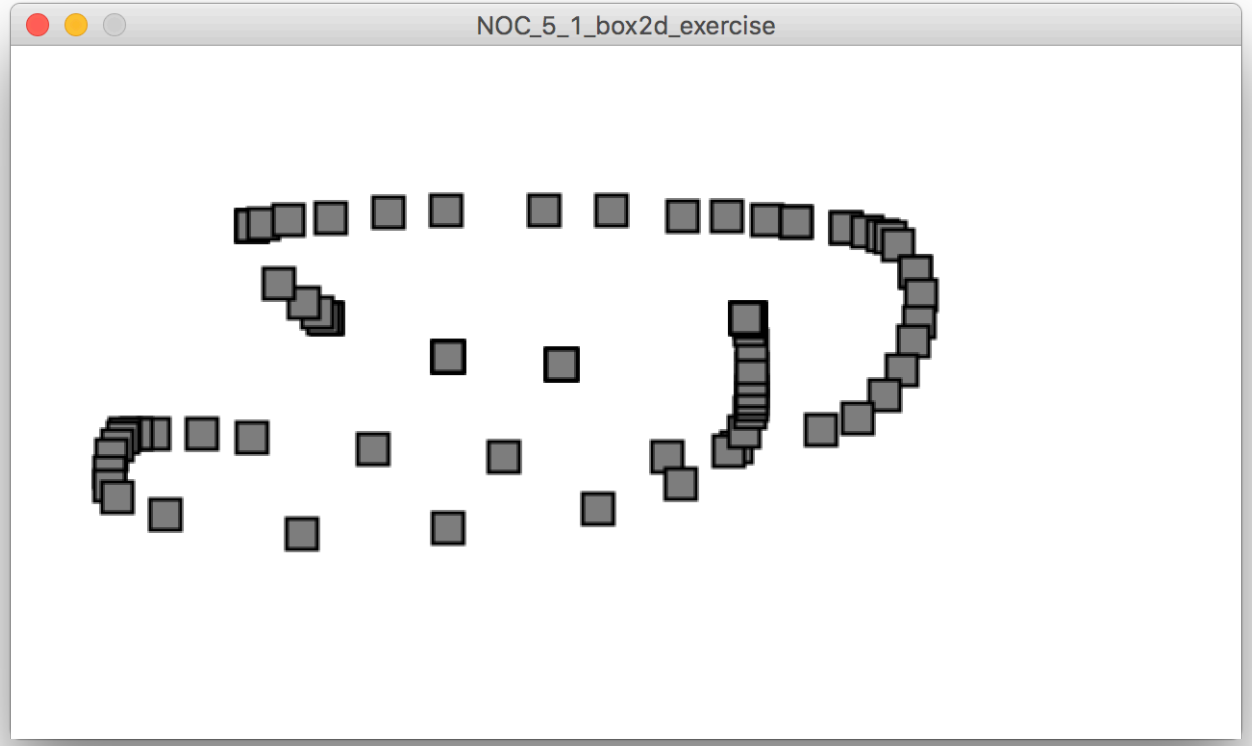


clockwise rotation in Processing

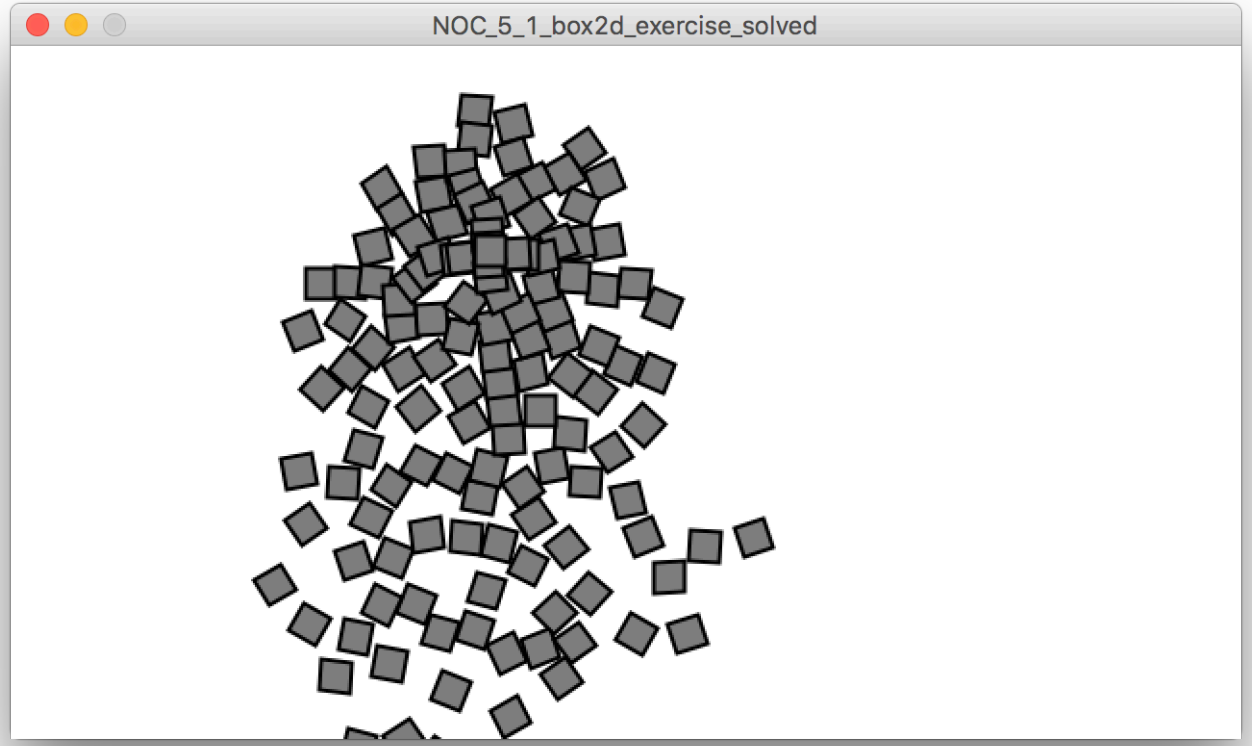


counterclockwise rotation in Box2D

Example



Example



Fixed Box2D Objects

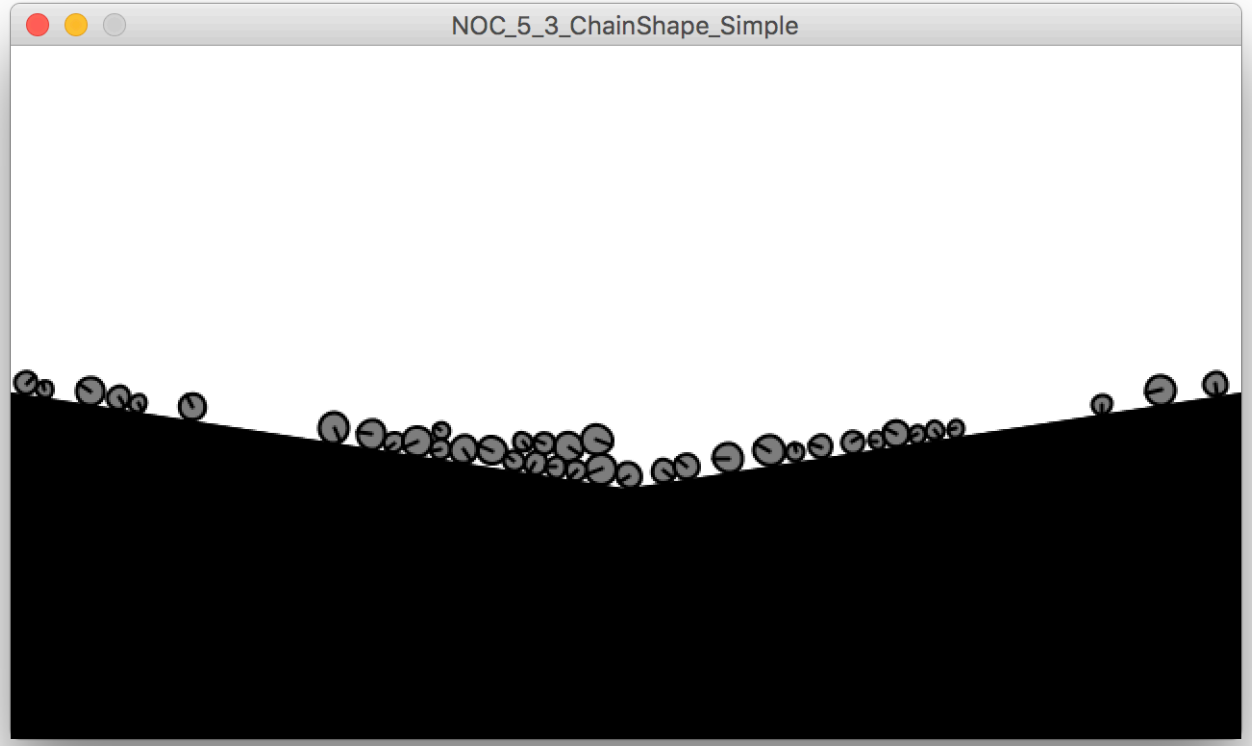
Fixed Box2D Objects

- `BodyDef bd = new BodyDef();`
- When BodyDef type = STATIC, the Body is locked in place.
- `bd.type = BodyType.STATIC;`

Fixed Box2D Objects

```
class Boundary {

  float x,y;
  float w,h;
  Body b;

  Boundary(float x_,float y_, float w_, float h_) {
    x = x_;
    y = y_;
    w = w_;
    h = h_;

    BodyDef bd = new BodyDef();
    bd.position.set(box2d.coordPixelsToWorld(x,y));
    bd.type = BodyType.STATIC;
    b = box2d.createBody(bd);

    float box2dW = box2d.scalarPixelsToWorld(w/2);
    float box2dH = box2d.scalarPixelsToWorld(h/2);
    PolygonShape ps = new PolygonShape();
    ps.setAsBox(box2dW, box2dH);

    b.createFixture(ps,1);
  }
  void display() {
    fill(0);
    stroke(0);
    rectMode(CENTER);
    rect(x,y,w,h);
  }

}
```

A Curvy Boundary



A Curvy Boundary

- ChainShape
- Step1
  - Define a body
    - BodyDef bd = new BodyDef();
    - Body body = box2d.world.createBody(bd);

A Curvy Boundary

- Step2
  - Define the Shape
    - ChainShape chain = new ChainShape();
- Step3
  - Configure the Shape
  - 곡선을 만들려면 점의 배열(Vec2) 필요
  - Vec2[] vertices = new Vec2[2];
  - vertices[0] = box2d.coordPixelsToWorld(0,150);
  - vertices[1] = box2d.coordPixelsToWorld(width,150);
  - **chain.createChain(vertices, vertices.length);**

A Curvy Boundary

- Step4
  - Attach the Shape to the body with a Fixture
  - FixtureDef fd = new FixtureDef();
  - **fd.shape = chain;**
  - fd.density = 1;
  - fd.friction = 0.3;
  - fd.restitution = 0.5;
  - body.createFixture(fd);

A Curvy Boundary

```
class Surface {
  ArrayList<Vec2>  surface;

  Surface() {

    surface = new ArrayList<Vec2>();
    surface.add(new Vec2(0, height/2+50));
    surface.add(new Vec2(width/2, height/2+50));
    surface.add(new Vec2(width, height/2));

    ChainShape chain = new ChainShape();

    Vec2[] vertices = new Vec2[surface.size()];


    for (int i = 0; i < vertices.length; i++) {
      vertices[i] = box2d.coordPixelsToWorld(surface.get(i));
    }

   chain.createChain(vertices, vertices.length);

    BodyDef bd = new BodyDef();
    Body body = box2d.world.createBody(bd);
    body.createFixture(chain, 1);
  }
```

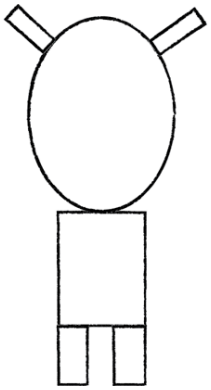
A Curvy Boundary

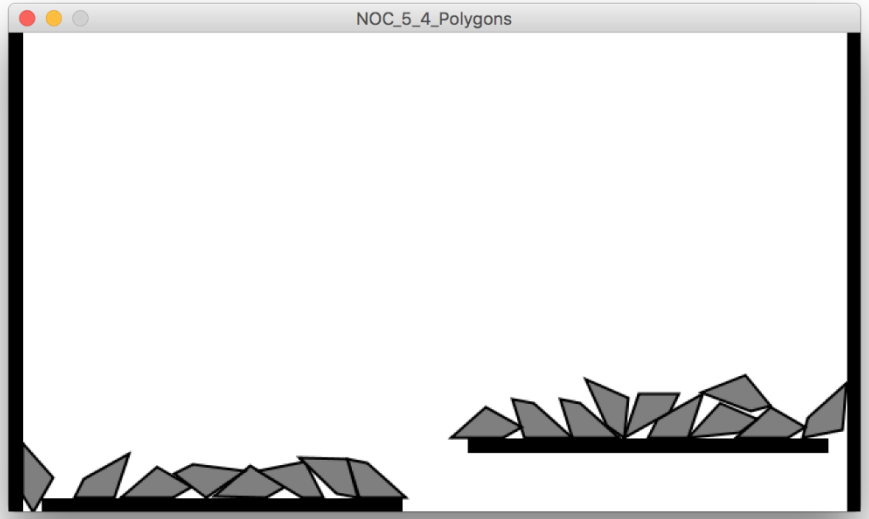
```
class Surface {
  ArrayList<Vec2>  surface;

  Surface() {

    surface = new ArrayList<Vec2>();
    surface.add(new Vec2(0, height/2+50));
    surface.add(new Vec2(width/2, height/2+50));
    surface.add(new Vec2(width, height/2));

    ChainShape chain = new ChainShape();

    Vec2[] vertices = new Vec2[surface.size()];


    for (int i = 0; i < vertices.length; i++) {
      vertices[i] = box2d.coordPixelsToWorld(surface.get(i));
    }

   chain.createChain(vertices, vertices.length);

    BodyDef bd = new BodyDef();
    Body body = box2d.world.createBody(bd);
    body.createFixture(chain, 1);
  }
```

A Curvy Boundary

```
 void display() {
    strokeWeight(1);
    stroke(0);
    noFill();
    //Draw the ChainShape as a series of vertices.
    beginShape();
    for (Vec2 v: surface) {
      vertex(v.x,v.y);
    }
    endShape();
  }
}
```

Complex Forms





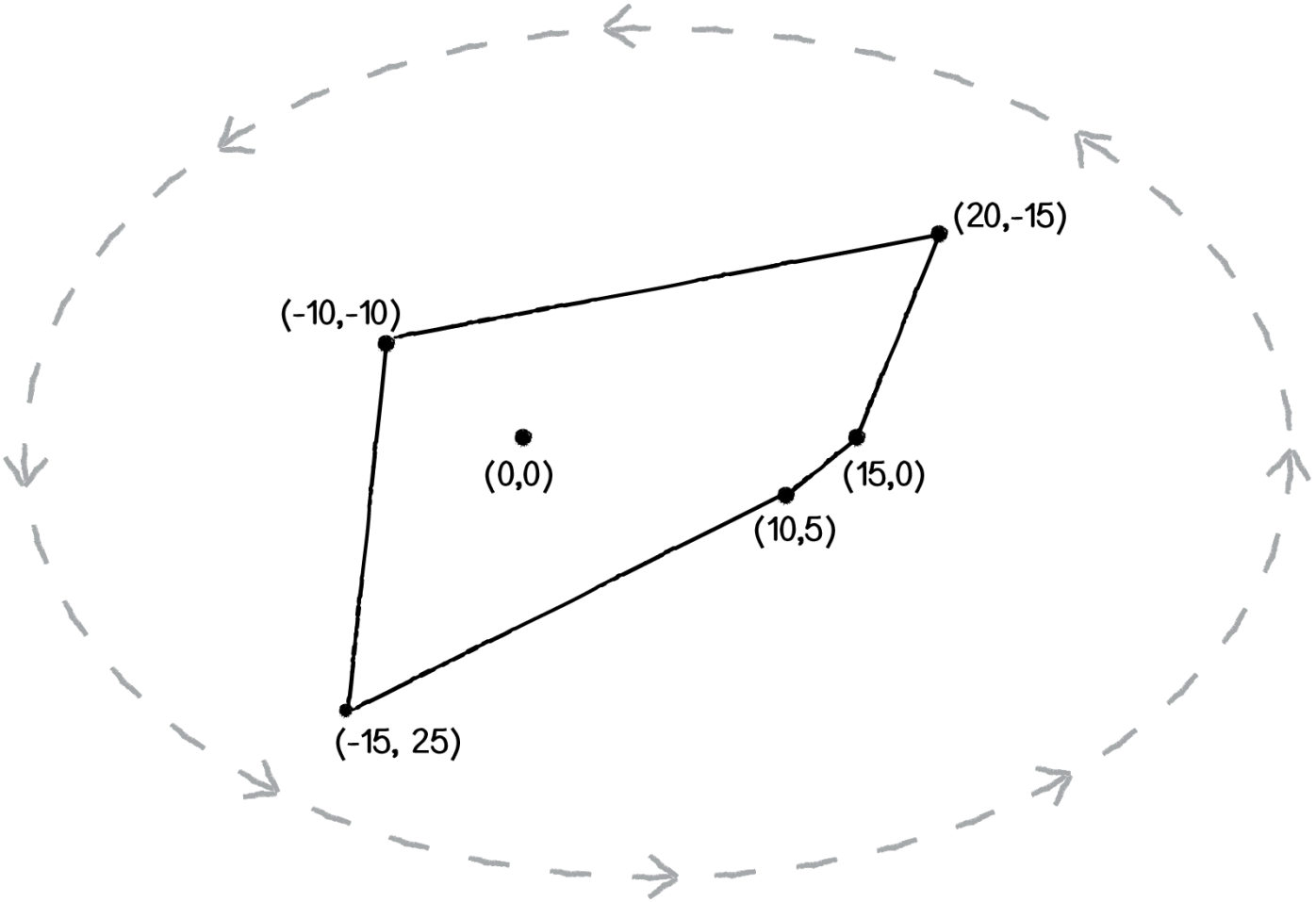
Complex Forms

- How to create more complex shapes?
- PolygonShape
- Connected Vertices

Complex Forms

```
Vec2[] vertices = new Vec2[4];  // An array of 4 vectors
vertices[0] = box2d.vectorPixelsToWorld(new Vec2(-15, 25));
vertices[1] = box2d.vectorPixelsToWorld(new Vec2(15, 0));
vertices[2] = box2d.vectorPixelsToWorld(new Vec2(20, -15));
vertices[3] = box2d.vectorPixelsToWorld(new Vec2(-10, -10));

PolygonShape ps = new PolygonShape();
ps.set(vertices, vertices.length);
```

Complex Forms

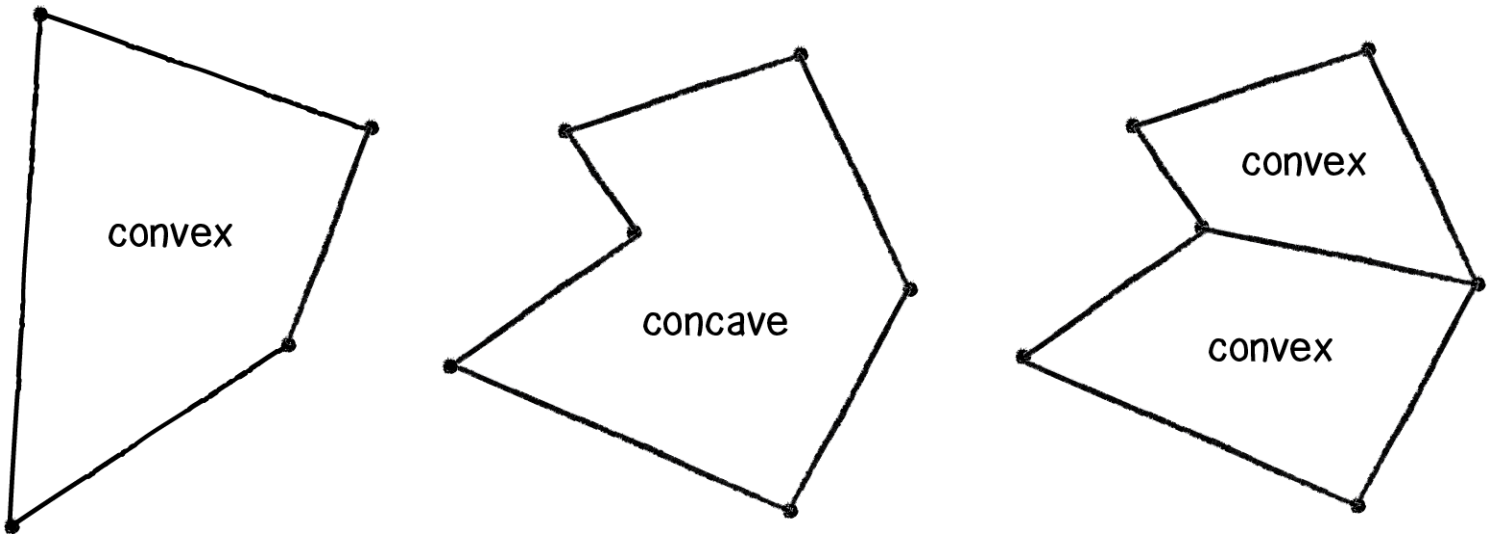


Complex Forms

- 점의 순서
  - 픽셀 좌표에서 점의 위치가 시계 반대 방향
  - Box2D는 시계 방향으로 만들어짐

- 볼록한 객체만 생성 가능
  - 오목한 모양은 충돌 처리가 안됨.

Complex Forms



Complex Forms

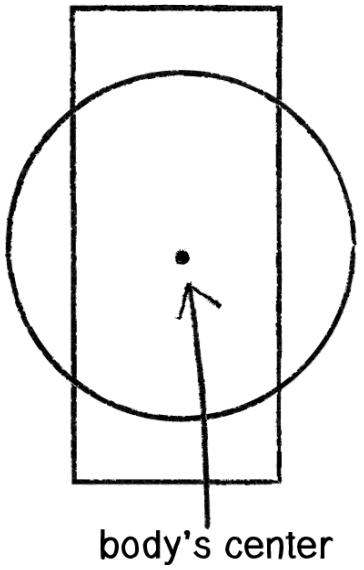
```
void display() {
    Vec2 pos = box2d.getBodyPixelCoord(body);
    float a = body.getAngle();

Fixture f = body.getFixtureList();
PolygonShape ps = (PolygonShape) f.getShape();

    rectMode(CENTER);
    pushMatrix();
    translate(pos.x,pos.y);
    rotate(-a);
    fill(175);
    stroke(0);
    beginShape();
for (int i = 0; i < ps.getVertexCount(); i++) {
      Vec2 v = box2d.vectorWorldToPixels(ps.getVertex(i));
      vertex(v.x,v.y);
    }
    endShape(CLOSE);
    popMatrix();
  }
```

Exercise 5.4

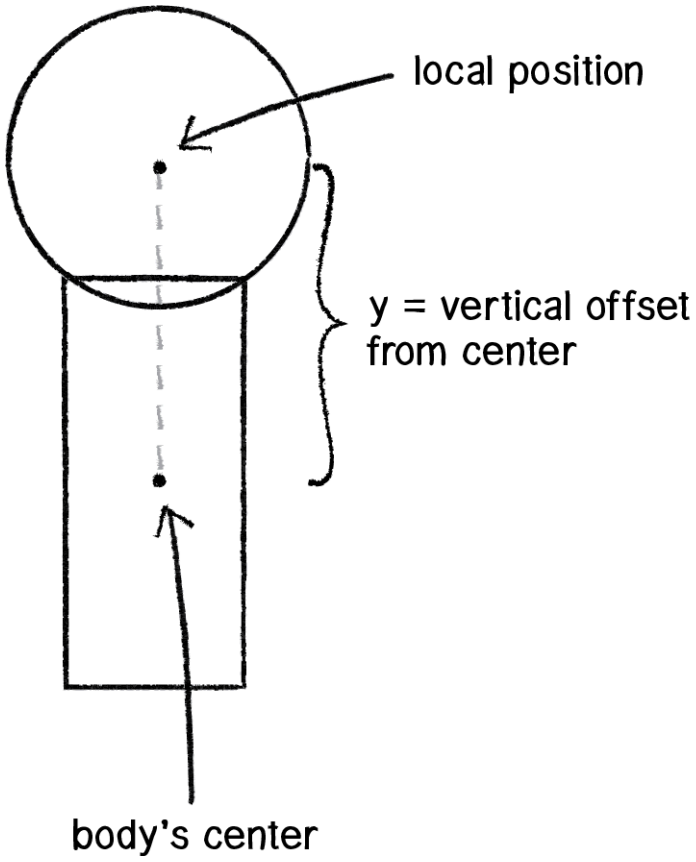
Complex Forms

- We can attach several shapes on a single body

```
BodyDef bd = new BodyDef();
bd.type = BodyType.DYNAMIC;
bd.position.set(box2d.coordPixelsToWorld(center));
body = box2d.createBody(bd);

Making shape 1 (the rectangle)
PolygonShape ps = new PolygonShape();
float box2dW = box2d.scalarPixelsToWorld(w/2);
float box2dH = box2d.scalarPixelsToWorld(h/2);
sd.setAsBox(box2dW, box2dH);

Making shape 2 (the circle)
CircleShape cs = new CircleShape();
cs.m_radius = box2d.scalarPixelsToWorld(r);

Attach both shapes with a fixture.
body.createFixture(ps,1.0);
body.createFixture(cs, 1.0);
```



Complex Forms

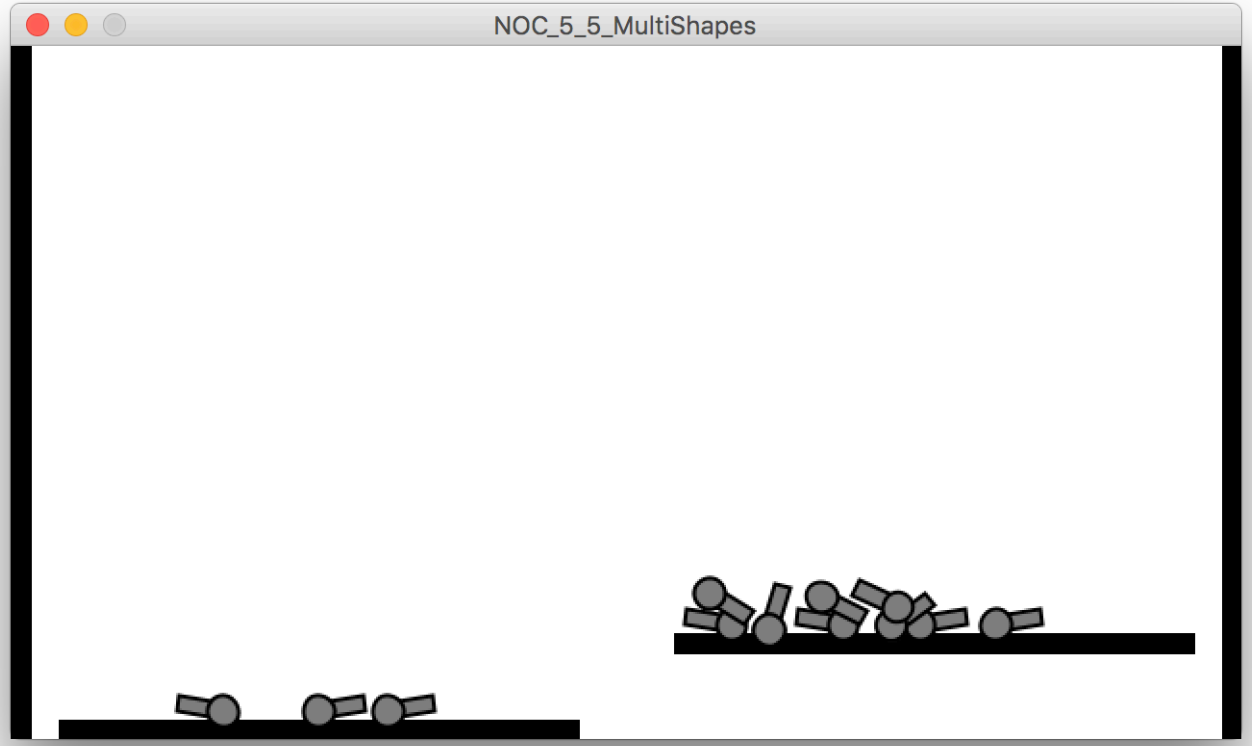


Vec2 offset = new Vec2(0,-h/2);

offset = box2d.vectorPixelsToWorld(offset);

circle.m_p.set(offset.x,offset.y);
m_p – local center of shape

Complex Shapes



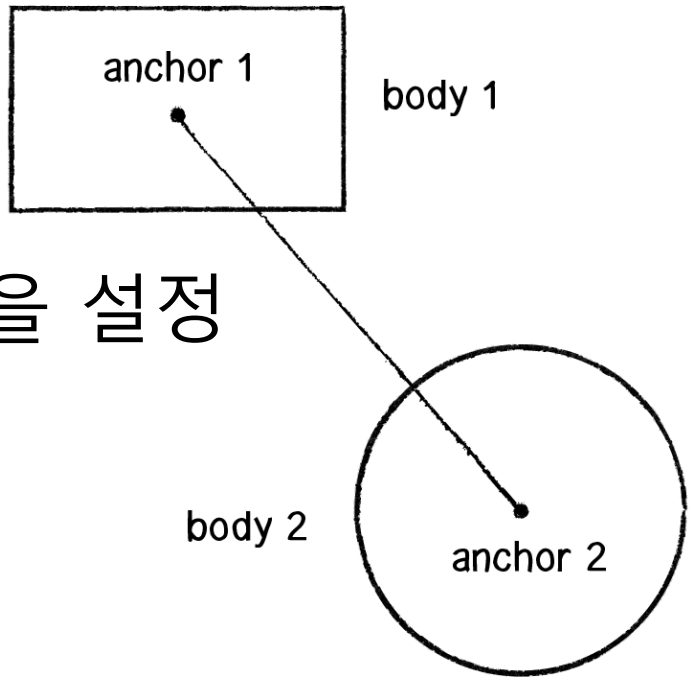
Box2D Joints

- Connecting one body to another.

- Three different joints
  - Distance joints
  - Revolute joints
  - Mouse joints

Distance Joints

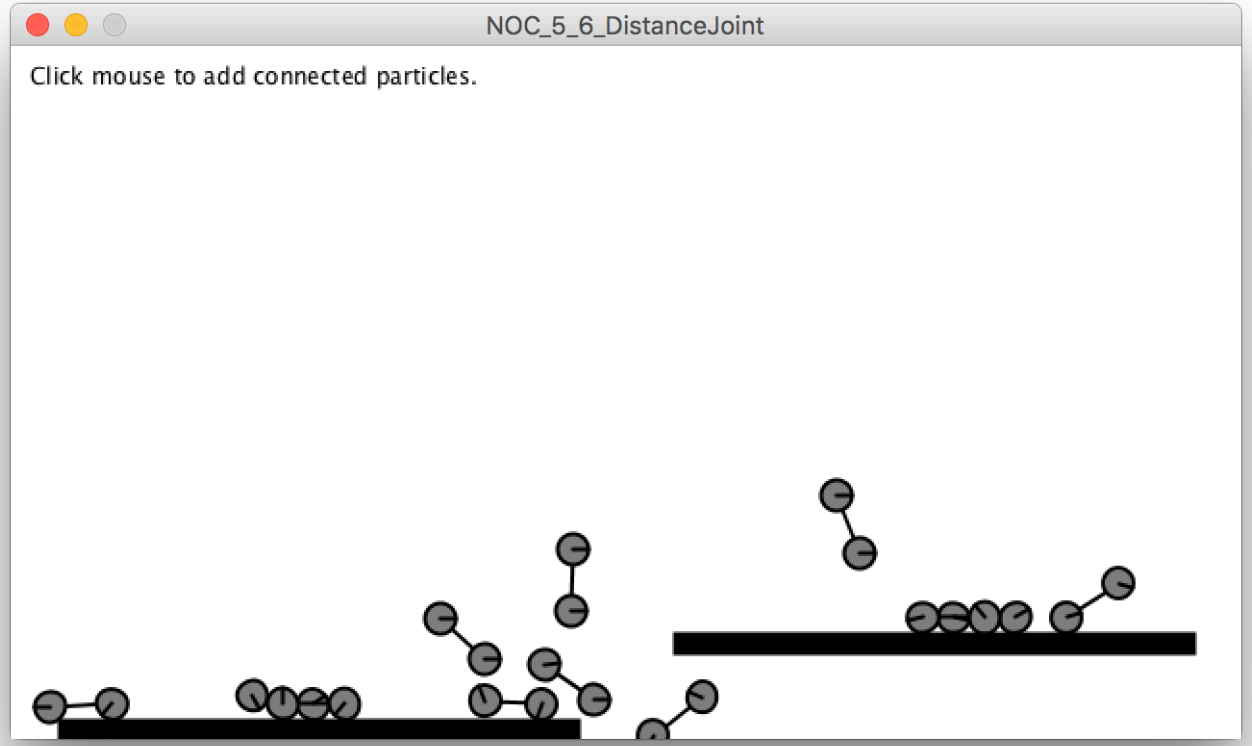
- **A joint that connects two bodies with a fixed length**
- Steps
  - 2개의 Body 객체를 준비
  - JointDef 객체 생성
  - 생성할 Joint 객체의 속성을 설정
  - Joint 객체를 생성



Distance Joints

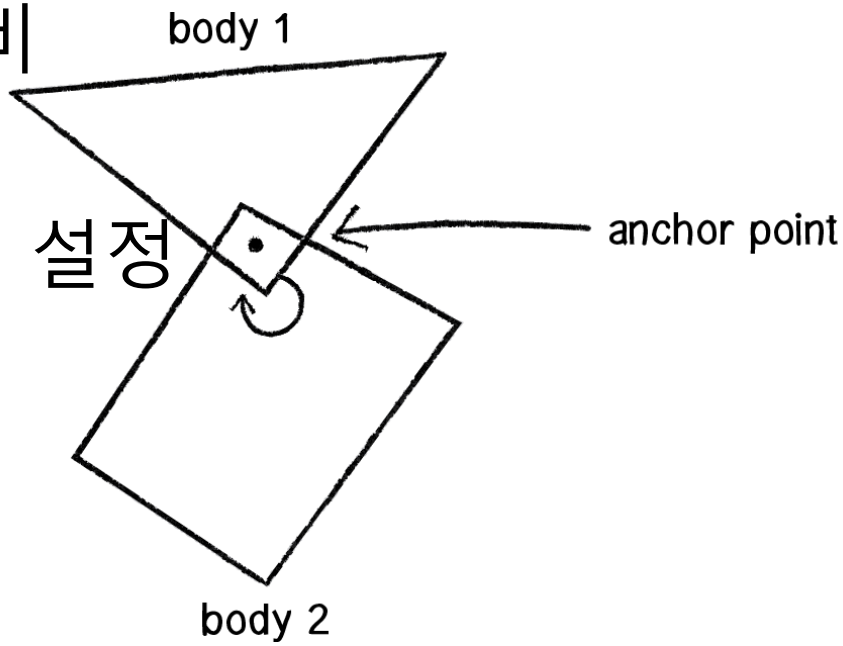
- Particle p1 = new Particle();
- Particle p2 = new Particle();
- DistanceJointDef djd = new DistanceJointDef();
- djd.bodyA = p1.body;
- djd.bodyB = p2.body;
- djd.length = box2d.scalarPixelsToWorld(10);
- **DistanceJoint dj = (DistanceJoint) box2d.world.createJoint(djd);**

Distance Joints

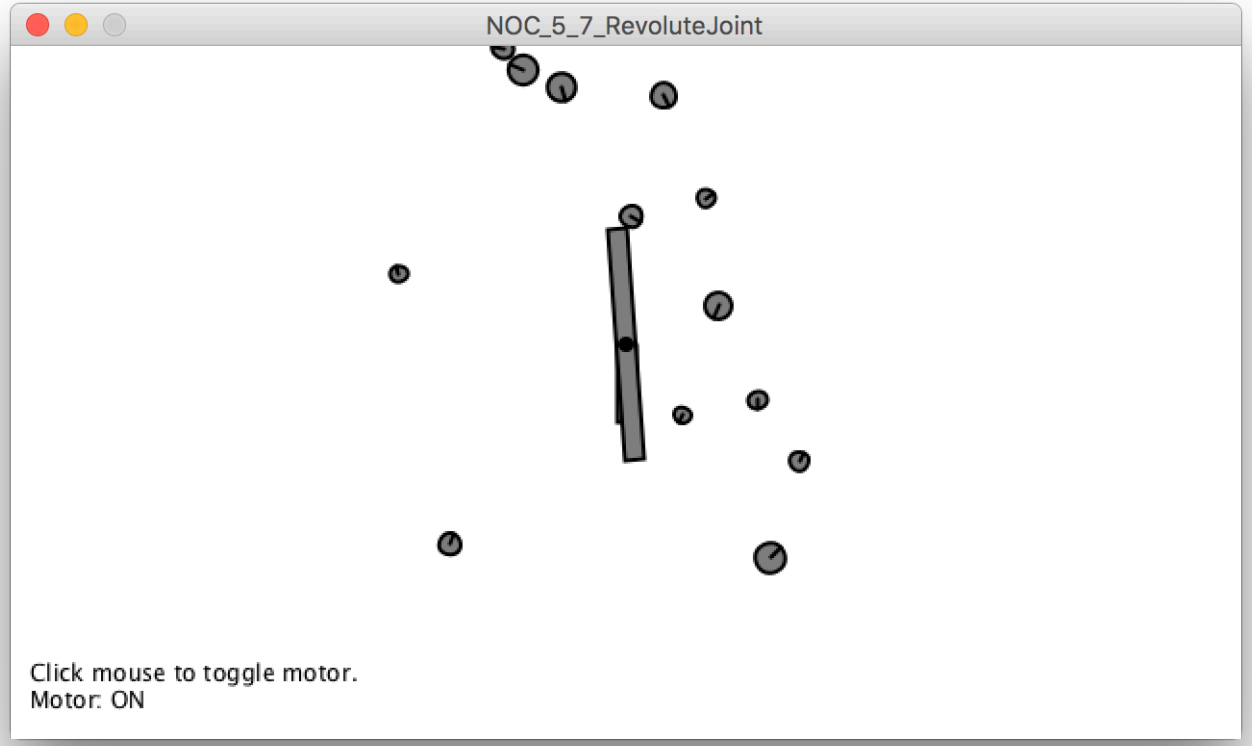


Revolute Joints

- **Connects two Box2D bodies at a common anchor point.**
- Steps
  - 2개의 Body 객체를 준비
  - JointDef 객체 생성
  - JointDef 객체의 속성을 설정
  - Joint 객체 생성



Revolute Joints



Revolute Joints

- Box box1 = new Box();
- Box box2 = new Box();
- RevoluteJointDef rjd = new RevoluteJointDef();
- **rjd.initialize(box1.body, box2.body, box1.body.getWorldCenter());**
- rjd.enableMotor = true;//turn on the motor
- rjd.motorSpeed = PI*2;//how fast?
- rjd.maxMotorTorque = 1000.0;//how powerful?
- rjd.enableLimit = true;
- rjd.lowerAngle = -PI/8;
- rjd.upperAngle = PI/8;
- **RevoluteJoint joint = (RevoluteJoint) box2d.world.createJoint(rjd);**

Make Windmill

```
class Windmill {

 //Our "Windmill" is two boxes and one joint.
  RevoluteJoint joint;
  Box box1;
  Box box2;

  Windmill(float x, float y) {

 //In this example, the Box class expects a boolean argument that will be
used to determine if //the Box is fixed or not.
     box1 = new Box(x,y,120,10,false);
     box2 = new Box(x,y,10,40,true);

     RevoluteJointDef rjd = new RevoluteJointDef();
     //The joint connects two bodies and is anchored at the center of the
first body.
     rjd.initialize(box1.body, box2.body, box1.body.getWorldCenter());


   //A motor!
     rjd.motorSpeed = PI*2;
     rjd.maxMotorTorque = 1000.0;
     rjd.enableMotor = true;

   //Create the Joint.
     joint = (RevoluteJoint) box2d.world.createJoint(rjd);
  }
```

```
Turning the motor on or off
  void toggleMotor() {
     boolean motorstatus = joint.isMotorEnabled();
     joint.enableMotor(!motorstatus);
  }

  void display() {
     box1.display();
     box2.display();
  }
}
```

Joints

http://www.iforce2d.net/b2dtut/joints-overview

revolute joint	기준 지점에 대한 회전 조인트
distance joint	각 바디에 대한 포인트가 고정된 거리 부분을 유지함
prismatic joint	두 바디의 상대적인 회전력이 고정되있다. 그리고 함께 축을 따라 미끄러진다
line joint	revolute 와 prismatic 의 조합, 차량 서스펜션 모델링에 유용하다
weld joint	같은 방향으로 두 바디를 고정한다
pully joint	로프와 도르래 방식의 조인트
friction joint	두 바디 사이의 상대적인 흔들림을 감소 시킨다
gear joint	두개의 조인트를 컨트롤 한다 그래서 한가지의 움직임이 다른 쪽에 영향을 미침.
mouse joint	Box2d world point로 바디를 잡아 당기는 것 (마우스/터치로 오브젝트를 당기는 데 사용함)

Applying Forces

```
class Box {
  Body body;

  void applyForce(Vec2 force) {
    Vec2 pos = body.getWorldCenter();
    //Calling the Body's applyForce() function
    body.applyForce(force, pos);
  }
}
```

Here we get to specify exactly where on the body the force is applied.

Collision Events

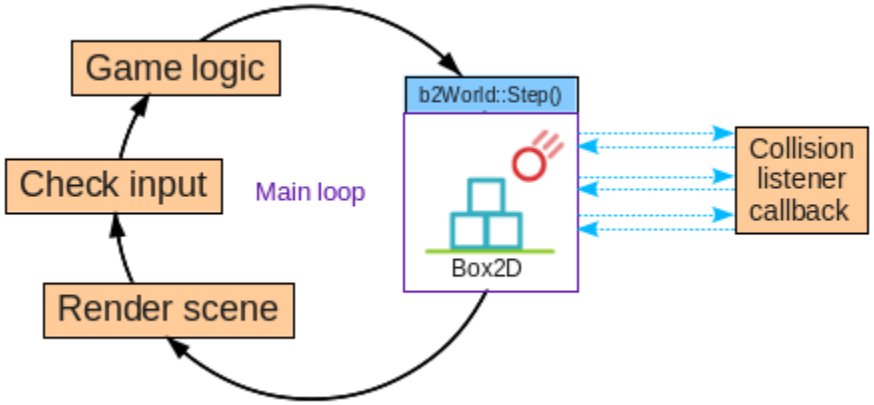
- Detecting collision events is done through a callback function
- **beginContact**() it triggered when two shapes collide.

Simple Interface Introuduction

```
interface Dot {
  void move();
  void display();
}

class CircleDot implements Dot {
  float x = 50;
  float y = 50;

  void move() {
    x = x + random(-1, 1);
  }

  void display() {
    ellipse(x, y, 16, 16);
  }
}
```

```
class SquareDot implements Dot {
  float x = 50;
  float y = 50;


  void move() {
    y = y + random(-1, 1);
  }

  void display() {
    rect(x, y, 16, 16);
  }
}
```

Collision callbacks



Collision callbacks

```
class MyContactListener : public b2ContactListener
  {
    void BeginContact(b2Contact* contact) {

      //check if fixture A was a ball
      void* bodyUserData = contact-
>GetFixtureA()->GetBody()->GetUserData();
      if ( bodyUserData )
        static_cast<Ball*>( bodyUserData )-
>startContact();

      //check if fixture B was a ball
      bodyUserData = contact->GetFixtureB()-
>GetBody()->GetUserData();
      if ( bodyUserData )
        static_cast<Ball*>( bodyUserData )-
>startContact();

    }
```

```
 void EndContact(b2Contact* contact) {

      //check if fixture A was a ball
      void* bodyUserData = contact-
>GetFixtureA()->GetBody()->GetUserData();
      if ( bodyUserData )
        static_cast<Ball*>( bodyUserData )-
>endContact();

      //check if fixture B was a ball
      bodyUserData = contact->GetFixtureB()-
>GetBody()->GetUserData();
      if ( bodyUserData )
        static_cast<Ball*>( bodyUserData )-
>endContact();

    }
  };
```

PBox2D Collision Detection

```
void setup() {
    box2d = new PBox2D(this);
    box2d.createWorld();
    Add this line if you want to listen for collisions.
    box2d.listenForCollisions();
}

void beginContact(Contact cp) {
    println("Something collided in the Box2D
World!");
}
```

Collision CallBack functions

1. beginContact() – 두 물체가 충동하는 시점
2. endContact() – 두 물체가 떨어지는 시점
3. preSolve() – 충동한다고 판단되는 연산을 할때 발생. 충돌이 일어나기 직전, 필요할 경우 충돌하지 않게 만들 수 있음.
4. postSolve() – 충돌과 관련 결과를 모두 연산한 후에 발생. 충돌과 관련된 충격량 등의 정보를 사용 가능

Collision

- Step1: Contact 객체로 Fixture 추출
  - Fixture f1 = cp.getFixtureA();
  - Fixture f2 = cp.getFixtureB();
- Step 2: Body 추출
  - Body b1 = f1.getBody();
  - Body b2 = f2.getBody();

Collision

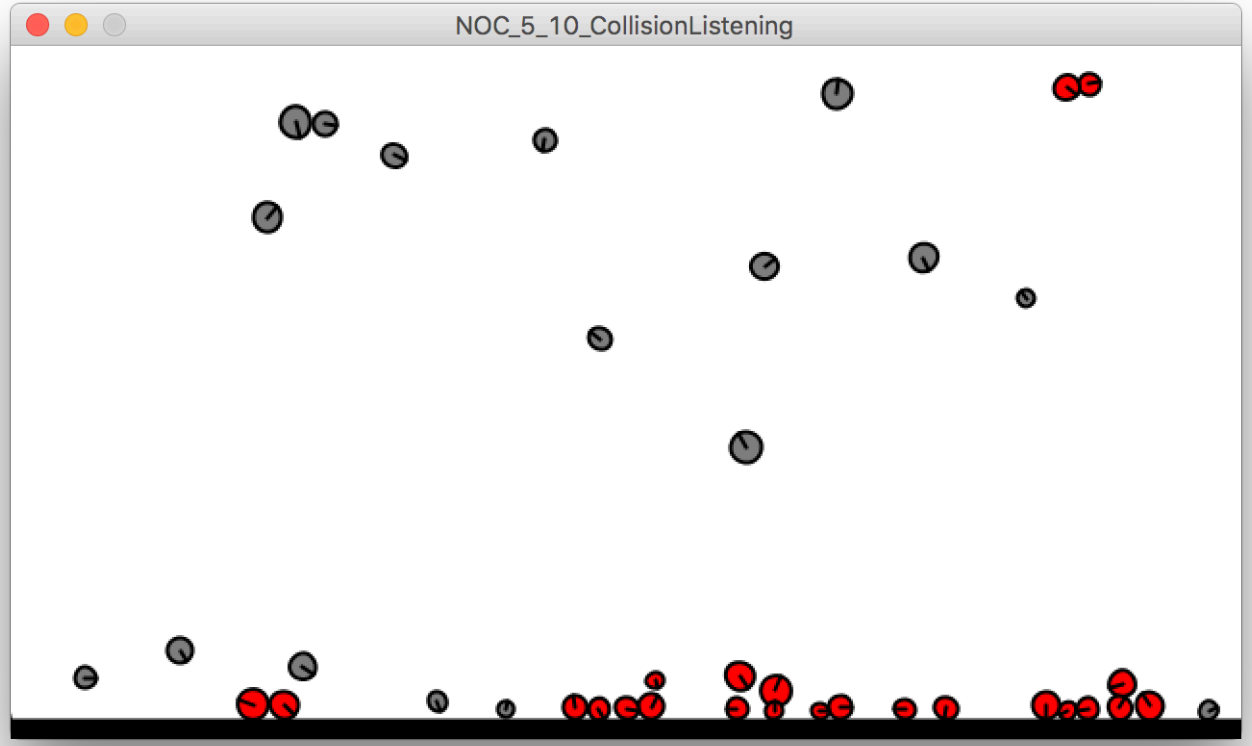
- Step 3: Body 객체로 어떤 Particle 객체인지 추출

```
class Particle {
  Body body;

  Particle(float x, float y, float r) {
    BodyDef bd = new BodyDef();
    bd.position = box2d.coordPixelsToWorld(x, y);
    bd.type = BodyType.DYNAMIC;
    body = box2d.createBody(bd);
    CircleShape cs = new CircleShape();
    cs.m_radius = box2d.scalarPixelsToWorld(r);
    body.createFixture(fd,1);

    body.setUserData(this);
  }
```

Collision CallBack

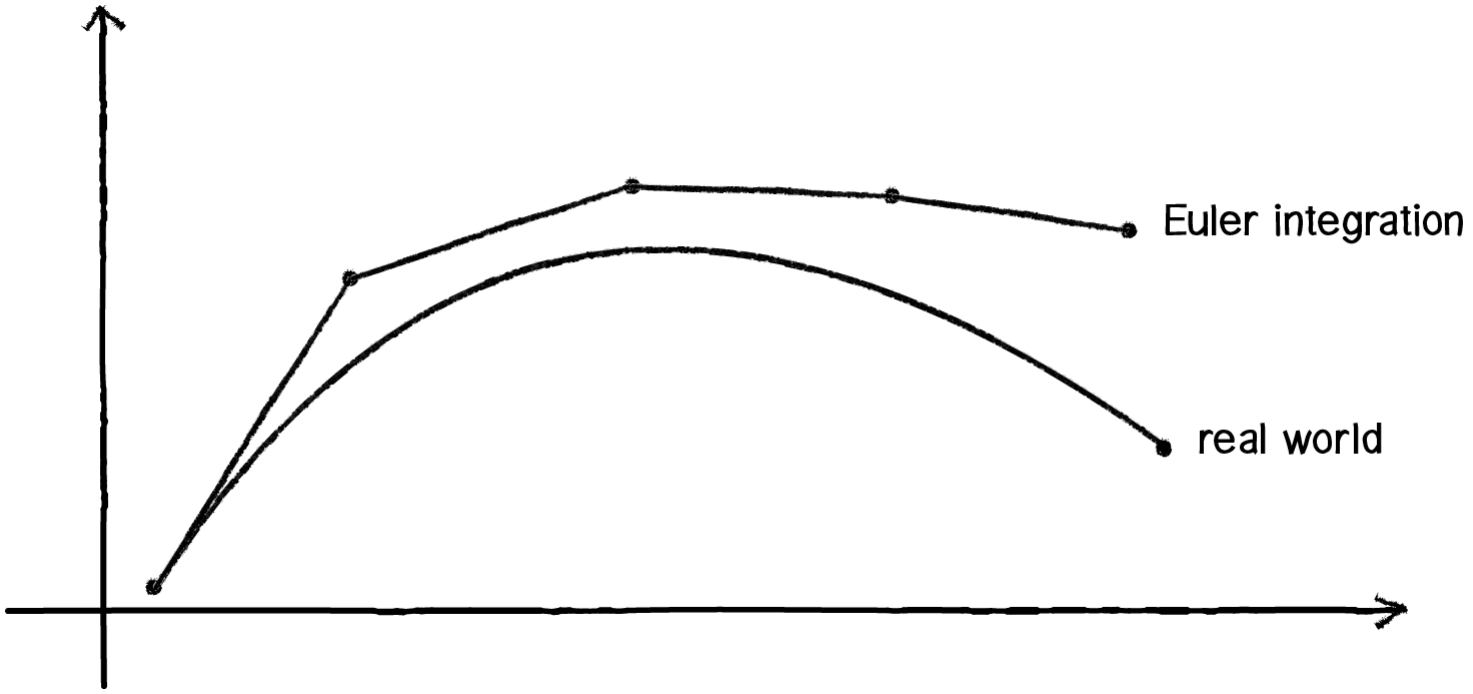
```
// Collision event functions!
void beginContact(Contact cp) {
  // Get both fixtures
  Fixture f1 = cp.getFixtureA();
  Fixture f2 = cp.getFixtureB();
  // Get both bodies
  Body b1 = f1.getBody();
  Body b2 = f2.getBody();

  // Get our objects that reference these bodies
  Object o1 = b1.getUserData();
  Object o2 = b2.getUserData();

  if (o1.getClass() == Particle.class && o2.getClass() == Particle.class) {
    Particle p1 = (Particle) o1;
    p1.change();
    Particle p2 = (Particle) o2;
    p2.change();
  }

}
```

Collision



Euler integration

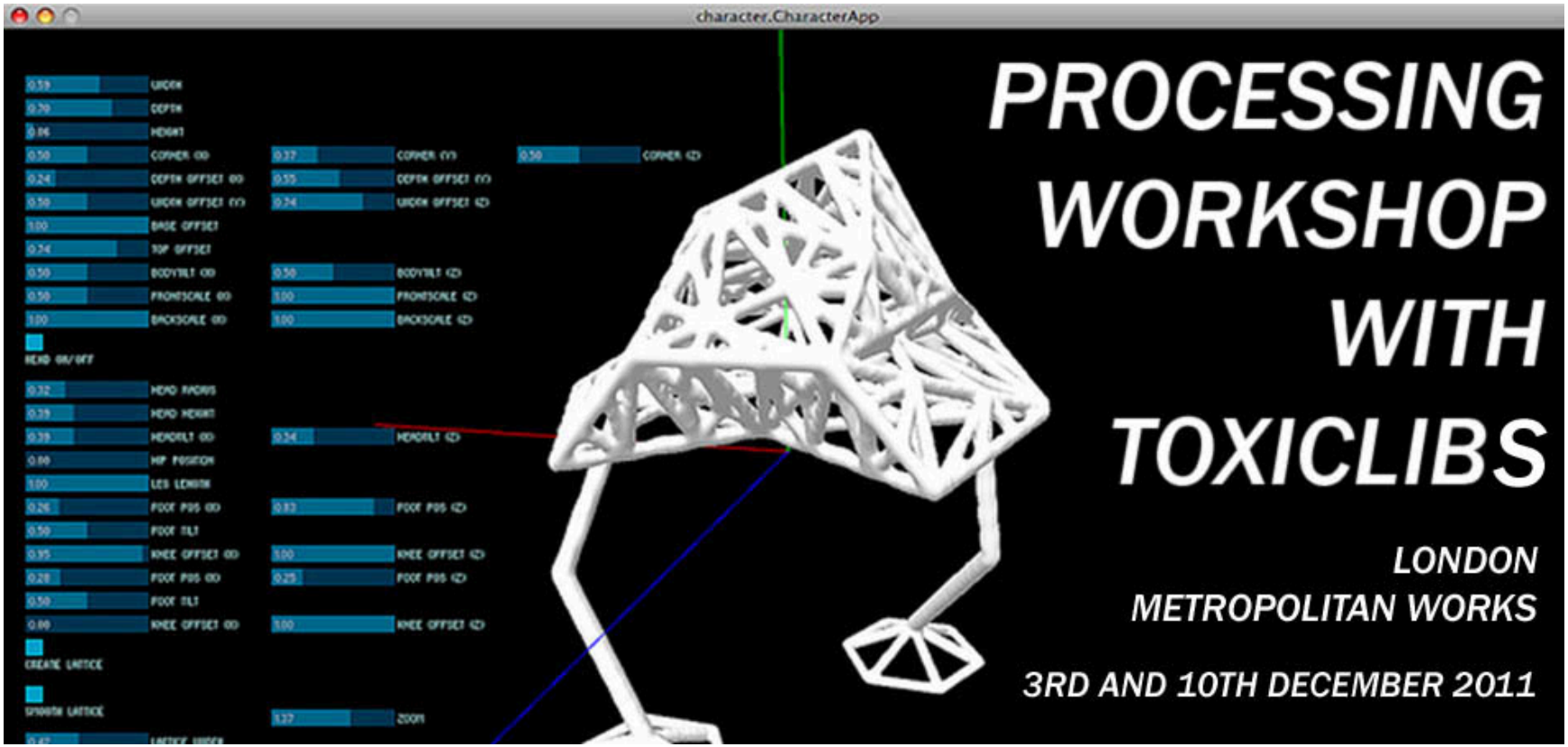
- Veloctiy.add(acceleration);
- Location.add(Veloctiy);



Better integration methods

- semi-explicit Euler : Box2D use
  - http://en.wikipedia.org/wiki/Symplectic_Euler_method
- Runge-Kutta : other game engine
- Verlet integration
  - Do not store velocity,
  - we always know the all past locations and current locations.
  - Library toxiclibs(http://toxiclibs.org/)

Toxiclib


character.CharacterApp
PROCESSING
WORKSHOP
WITH
TOXICLIBS
LONDON
METROPOLITAN WORKS
3RD AND 10TH DECEMBER 2011

Toxiclib

- toxiclibs includes a suite of other wonderful packages that help with audio, color, geometry, and more
- designed specifically for use with Processing
  - coordinate system that we'll use for the physics engine is the coordinate system of Processing
  - all of the physics simulations and functions work in both two and three dimensions
- Does not have collision detection
- Best work if you have lots of particles flying around the screen. Sometimes they attract each other. Sometimes they repel each other. And sometimes they are connected with springs.
  - High performance due to the Verlet integration algorithm

Toxiclib

- toxi.audio
- toxi.color
- toxi.geom
- toxi.math
- toxi.physics
- toxi.image.util
- toxi.util.datatypes

Box2D vs Toxiclib

Feature	Box2D	toxiclibs Verlet Physics
Collision geometry	Yes	No
3D physics	No	Yes
Particle attraction / repulsion forces	No	Yes
Spring connections	Yes	Yes
Other connections: revolute, pulley, gear, prismatic	Yes	No
Motors	Yes	No
Friction	Yes	No

Verlet Physics

- toxiclibs (http://toxiclibs.org/)
- core elements

Box2D	toxiclibs VerletPhysics
World	VerletPhysics
Body	VerletParticle
Shape	Nothing! toxiclibs does not handle shape geometry
Fixture	Nothing! toxiclibs does not handle shape geometry
Joint	VerletSpring

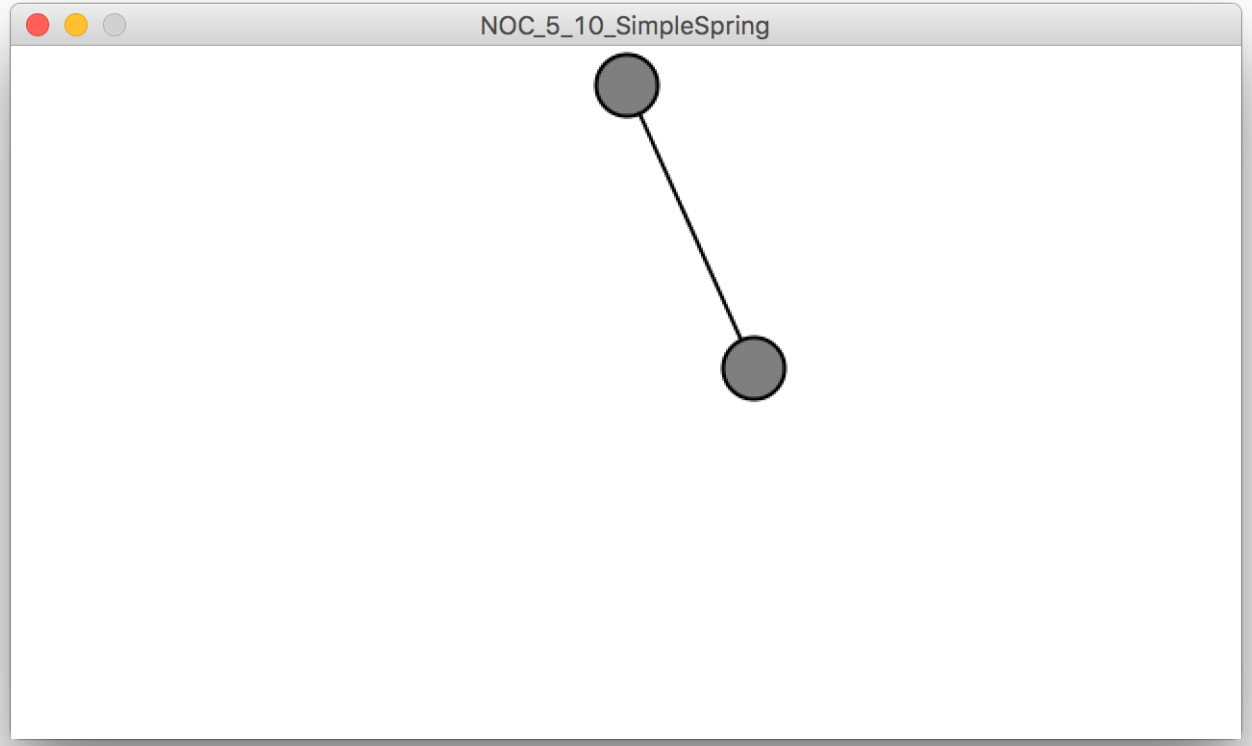
New Vector

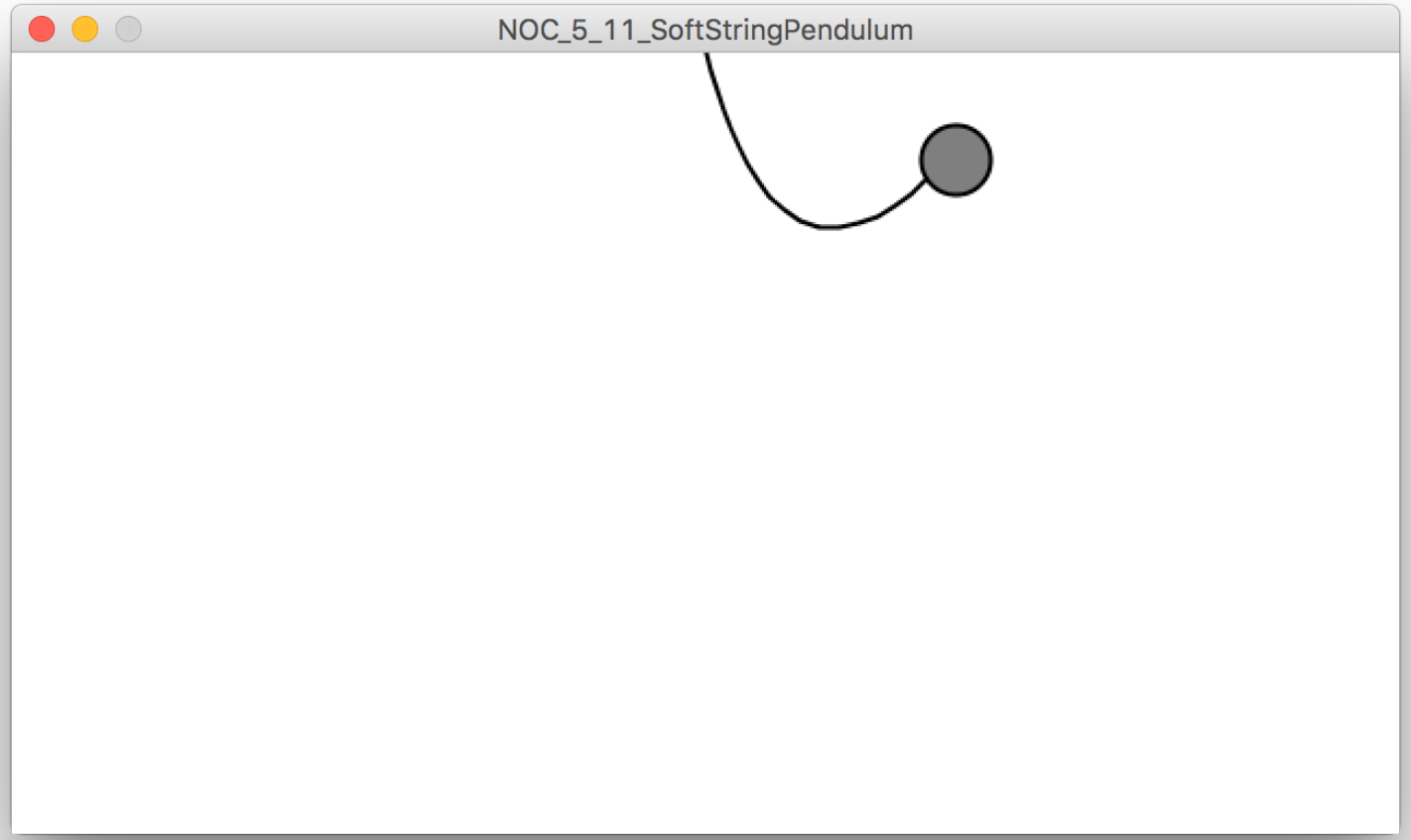
- Vec2D (http://toxiclibs.org/docs/core/toxi/geom/Vec2D.html)
- Vec3D (http://toxiclibs.org/docs/core/toxi/geom/Vec3D.html)

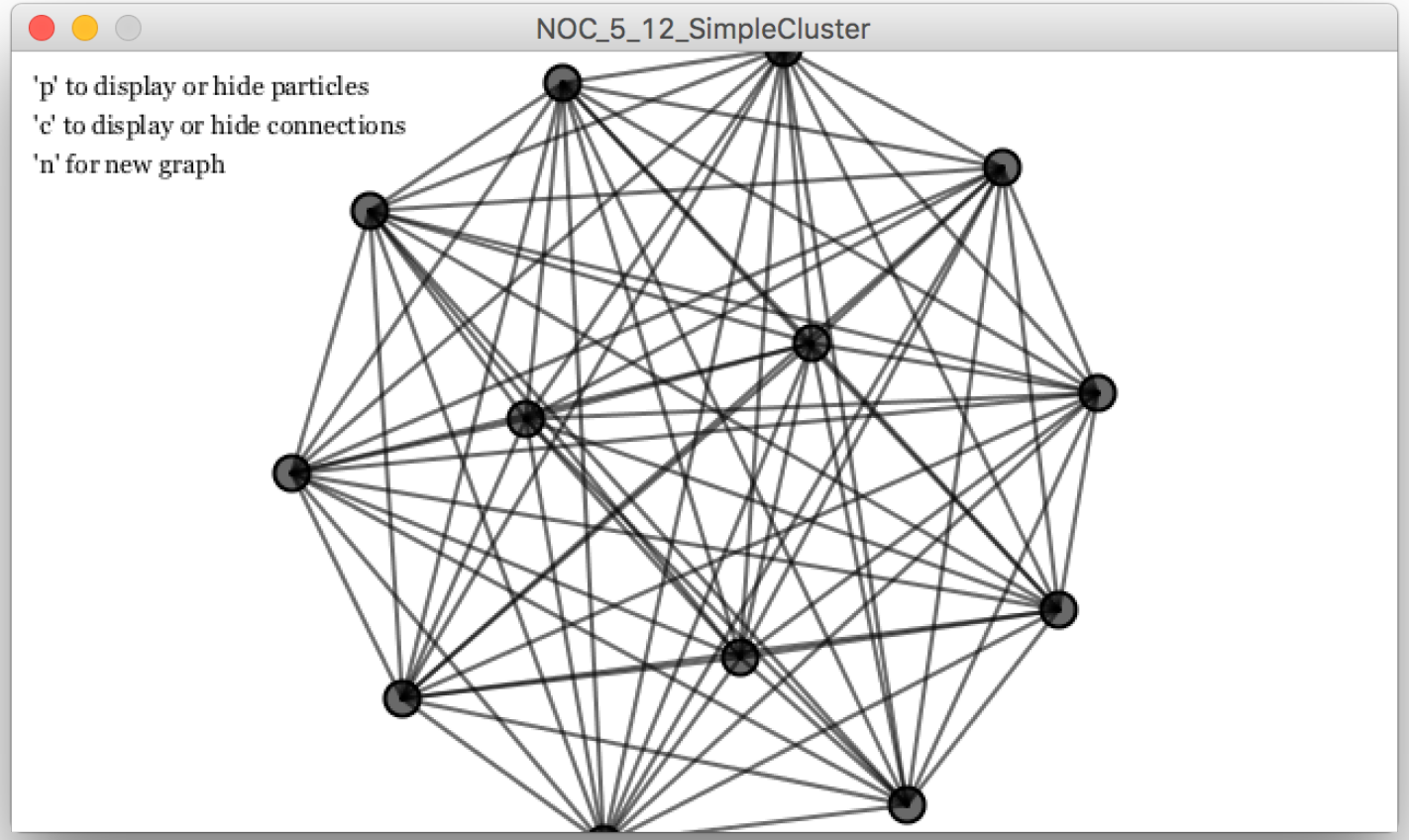
PVector	Vec2D
PVector a = new PVector(1,-1); PVector b = new PVector(3,4); a.add(b);	Vec2D a = new Vec2D(1,-1); Vec2D b = new Vec2D(3,4); a.addSelf(b);
PVector a = new PVector(1,-1); PVector b = new PVector(3,4); PVector c = PVector.add(a,b);	Vec2D a = new Vec2D(1,-1); Vec2D b = new Vec2D(3,4); Vec2D c = a.add(b);

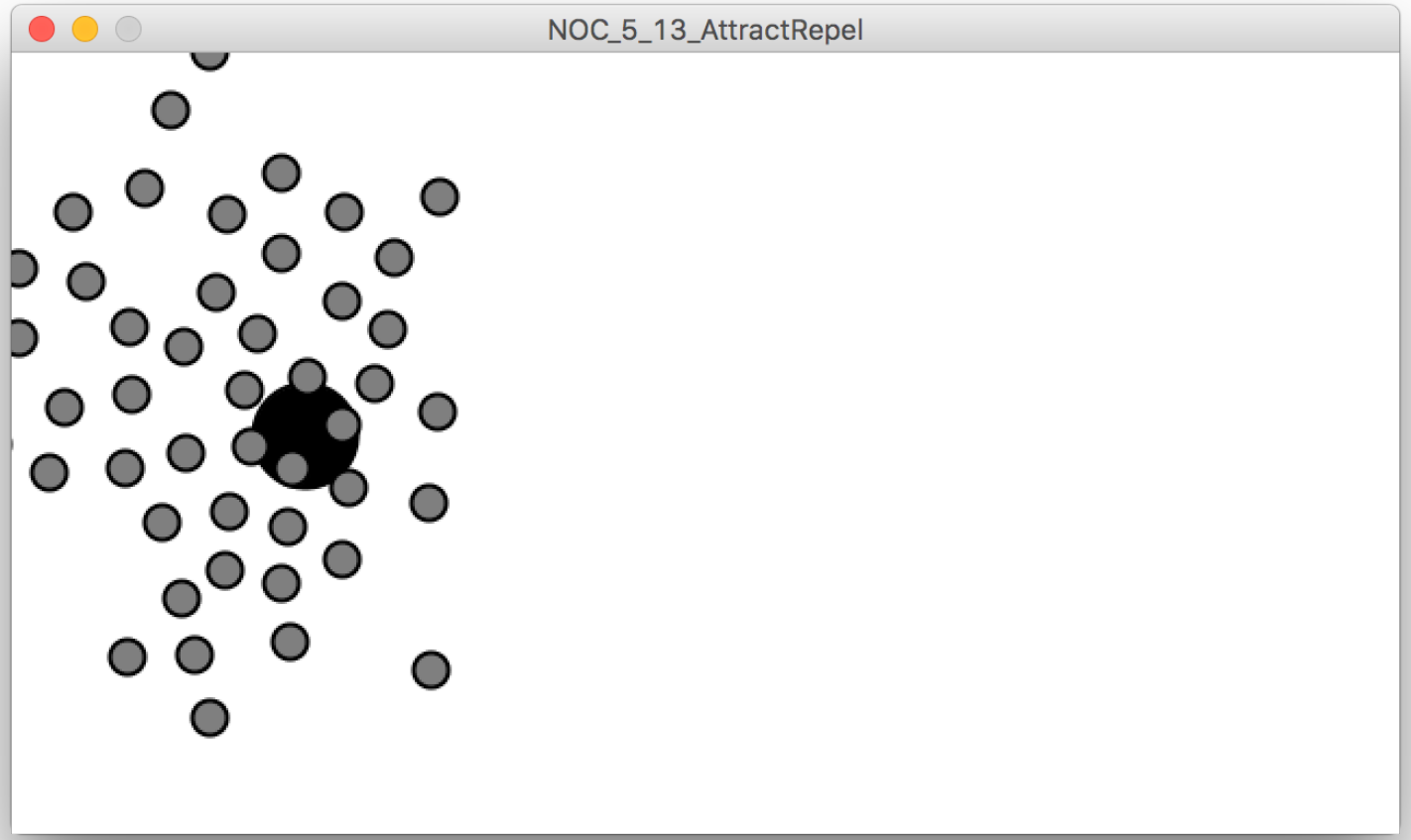
PVector a = new PVector(1,-1);
float m = a.mag();
a.normalize();
	Vec2D a = new Vec2D(1,-1);
	float m = a.magnitude();
	a.normalize();

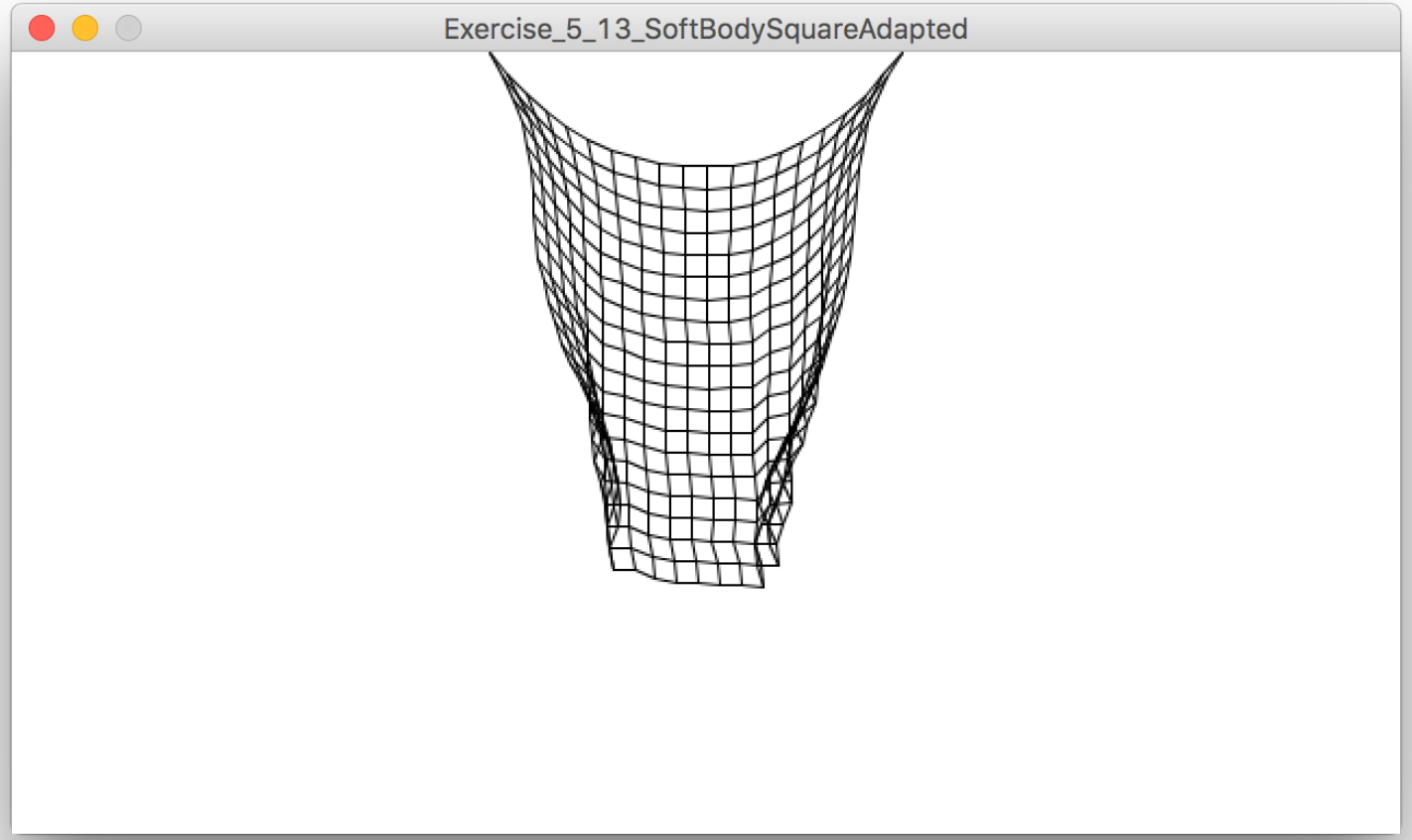
Example

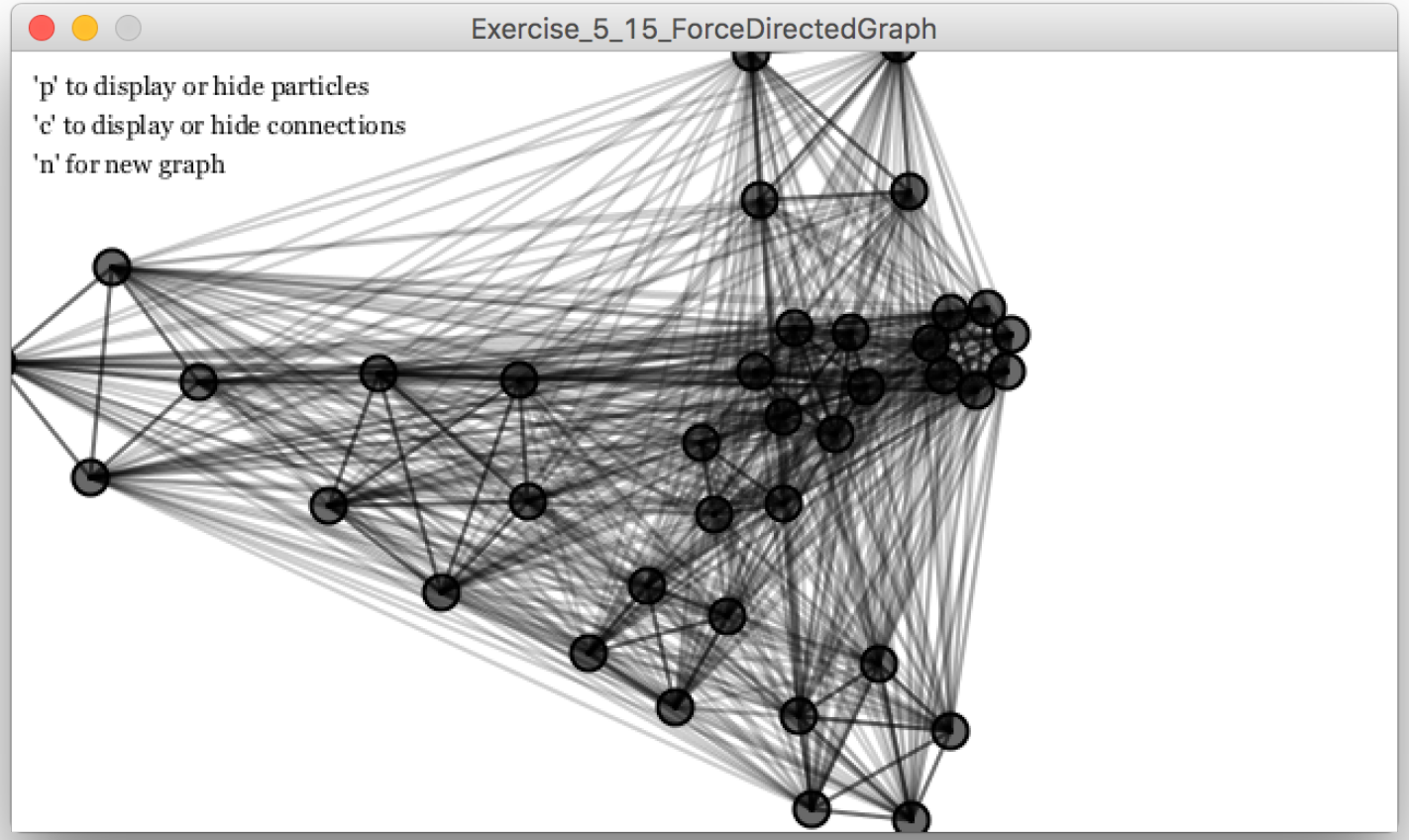



NOC_5_11_SoftStringPendulum


NOC_5_12_SimpleCluster
'p' to display or hide particles
'c' to display or hide connections
'n' for new graph


NOC_5_13_AttractRepel


Exercise_5_13_SoftBodySquareAdapted


Exercise_5_15_ForceDirectedGraph
'p' to display or hide particles
'c' to display or hide connections
'n' for new graph

End….

- Q&A