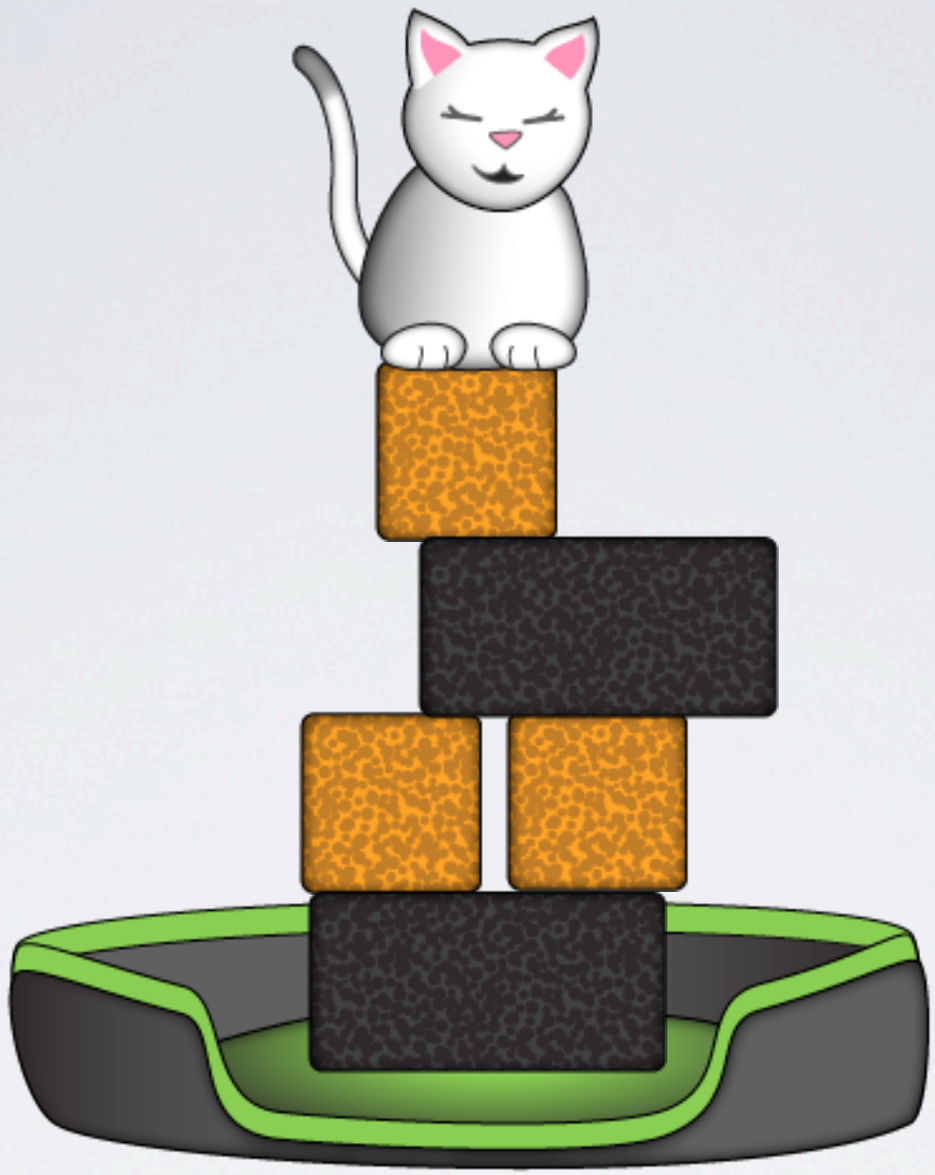
Twitter Hashtag: #cvm



COCOS2D VIA MINIGAMES

Basic Chipmunk Physics

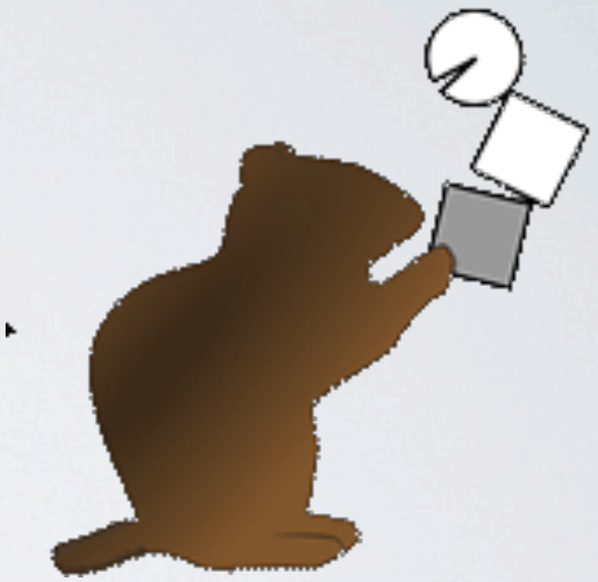
Cocos2DBook.com

SECTION 4 - PART I

- What is Chipmunk?
- Chipmunk vs. Box2D
- How Chipmunk Works
- How To Create a Basic Chipmunk Scene
- Demo: Hello, Chipmunk!

WHAT IS CHIPMUNK?

- An easy-to-use physics library
- Included with Cocos2D
- C-based API
- Based on Box2D
- All code included!



CHIPMUNK VS BOX2D

- C-based
- Units: points
- Fairly terse
- Constraints
- No Continuous Collision Detection

- C++ based
- Units: meters
- More verbose
- Joints
- Continuous Collision Detection

HOW CHIPMUNK WORKS

- Chipmunk has a "virtual space" to simulate physics
- Your job:
  - Add bodies and shapes
  - Specify gravity and other forces
- Every update:
  - Step sim
  - Draw scene based on results

gravity

wind

tractor beam

Friday, June 10, 2011

CHIPMUNK SPACE

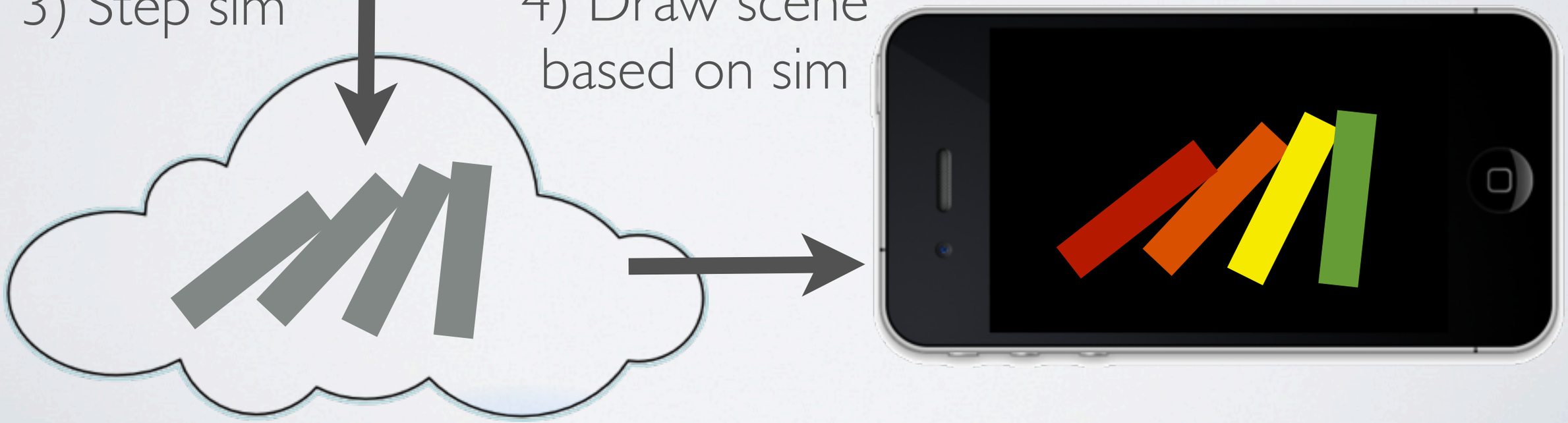
COCOS2D SCENE

1) Add shapes, forces

2) Draw scene based on sim

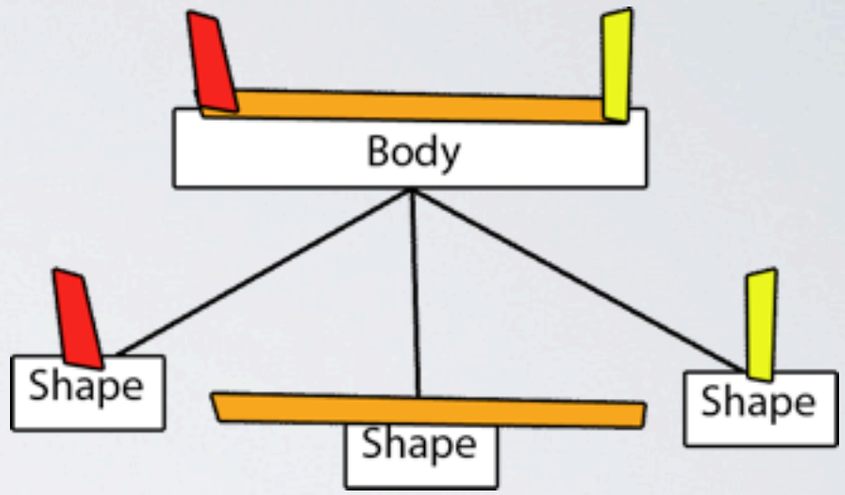
3) Step sim

4) Draw scene based on sim



BODIES AND SHAPES

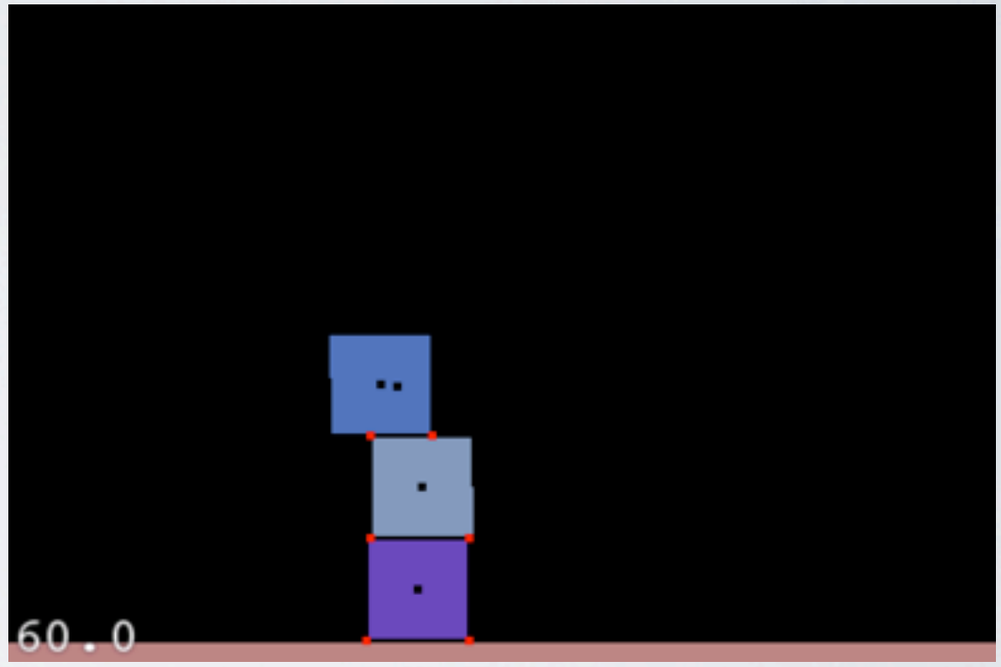
- Bodies have one or more shapes
- Bodies have mass



- Two different types of bodies:
  - Dynamic
  - Static
- Shapes have properties
  - elasticity, friction
- Different kinds of shapes
  - Box, Circle, Poly, Segment

HOW TO CREATE A BASIC CHIPMUNK SCENE

1. Initialize Chipmunk (just once!)
2. Create Chipmunk "space"
3. (Optional) Add "ground"
4. (Optional) Add bodies/shapes
5. Step sim in update loop
6. (Optional) Enable debug draw
7. (Optional) Add mouse joints



1. INITIALIZE CHIPMUNK

```
#import "chipmunk.h"
```

```
cpInitChipmunk();
```

2. CREATE CHIPMUNK "SPACE"

```
cpSpace *space = cpSpaceNew();
space->gravity = ccp(0, -750);
cpSpaceResizeStaticHash(space, 400, 200);
cpSpaceResizeActiveHash(space, 200, 200);
```

- Gravity - global force
- Hash: optimization for quick collision detection
  - Recommended size: > average object
  - Recommended cells: 10x number objects

3. (OPTIONAL) ADD "GROUND"

```
cpBody * groundBody = cpBodyNewStatic();

float radius = 10.0;
cpShape *groundShape = cpSegmentShapeNew
 (groundBody, lowerLeft, lowerRight, radius);
groundShape->e = 0.5; // elasticity
groundShape->u = 1.0; // friction
cpSpaceAddShape(space, groundShape);
```

4. (OPTIONAL) ADD BODIES/SHAPES

```
float boxSize = 60.0;
float mass = 1.0;
cpBody *body = cpBodyNew(mass,
 cpMomentForBox(mass, boxSize, boxSize));
body->p = location;
cpSpaceAddBody(space, body);

cpShape *shape =
 cpBoxShapeNew(body, boxSize, boxSize);
shape->e = 1.0;
shape->u = 1.0;
cpSpaceAddShape(space, shape);
```

5. STEP SIM IN UPDATE LOOP

```
int steps = 2;
CGFloat stepDt = dt/(CGFloat)steps;
for(int i=0; i<steps; i++){
    cpSpaceStep(space, stepDt);
}
```

- More steps = more accuracy
- Fixed-rate timeloop better - see book or sample code

6. (OPTIONAL) ENABLE DEBUG DRAW

- Need drawSpace.c, drawSpace.h to your project

- Add to draw method:

```
drawSpaceOptions options = {
        0, // drawHash
        0, // drawBBs,
        1, // drawShapes
        4.0, // collisionPointSize
        4.0, // bodyPointSize,
        2.0 // lineThickness
};

drawSpace(space, &options);
```

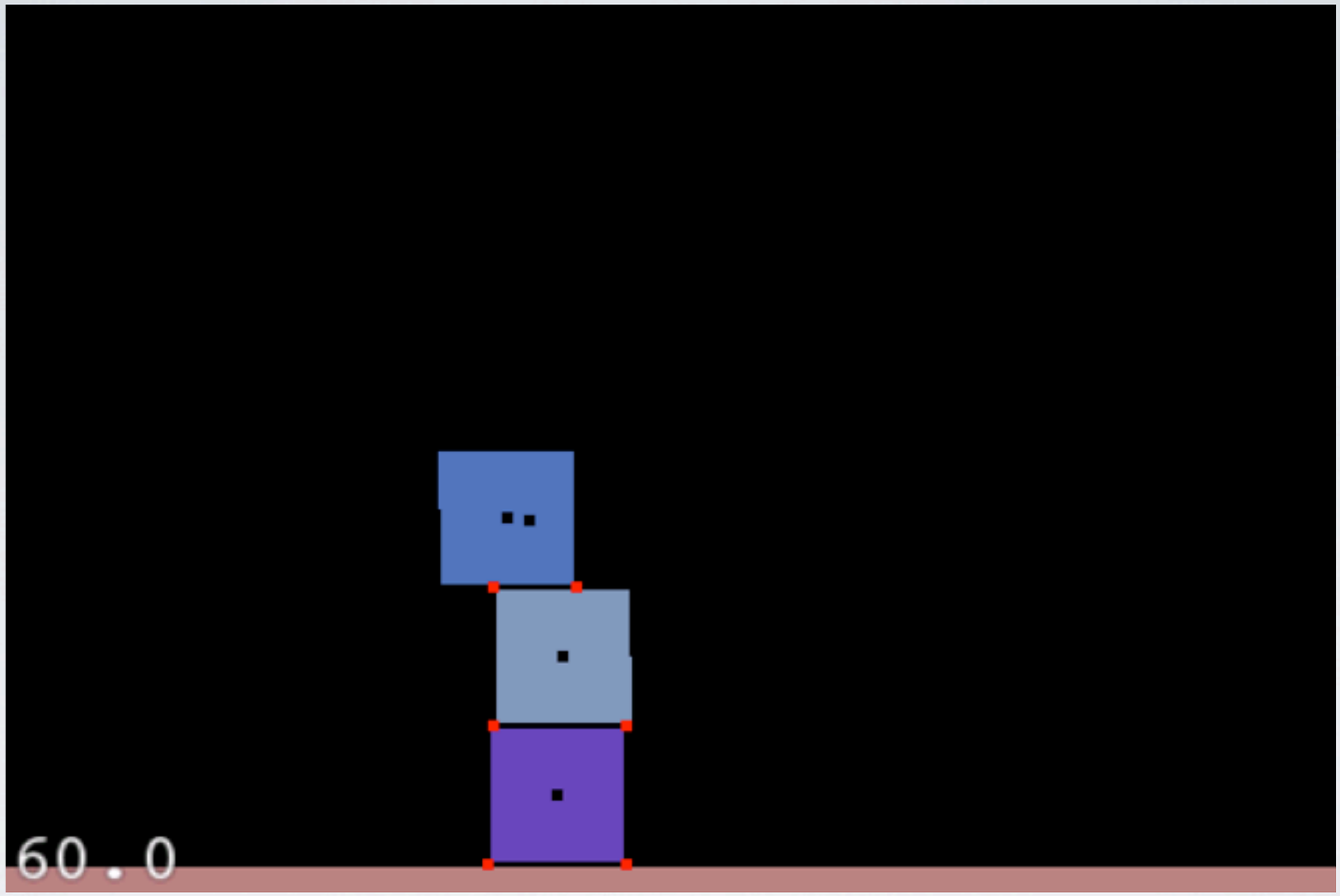
7. (OPTIONAL) ADD MOUSE JOINTS

- Add cpMouse.c, cpMouse.h to project
- In init: `mouse = cpMouseNew(space);`
- Add touch handlers:

```
- (void)ccTouchMoved:(UITouch *)touch withEvent:(UIEvent *)event {
    CGPoint touchLocation = [self convertTouchToNodeSpace:touch];
    cpMouseMove(mouse, touchLocation);
}

- (void)ccTouchEnded:(UITouch *)touch withEvent:(UIEvent *)event {
    cpMouseRelease(mouse);
}

- (void)ccTouchCancelled:(UITouch *)touch withEvent:(UIEvent *)event {
    cpMouseRelease(mouse);
}
```



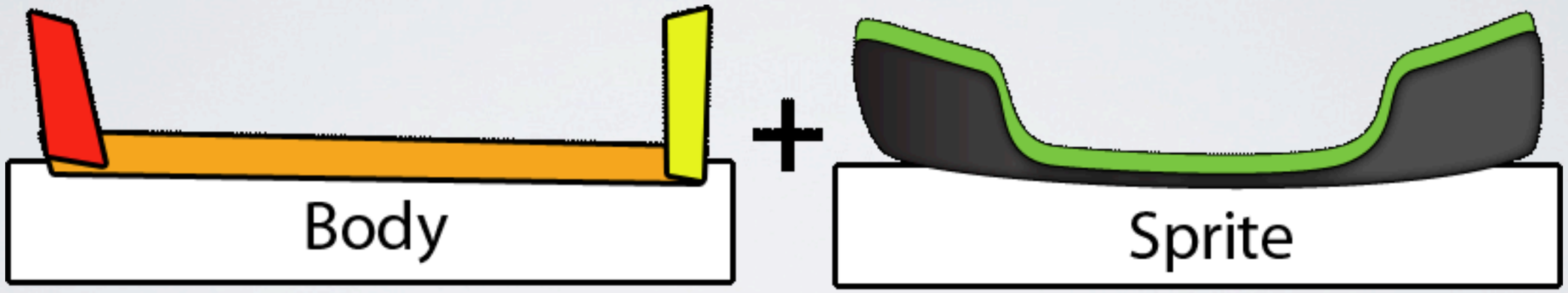
CODE+DEMO

Hello, Chipmunk!

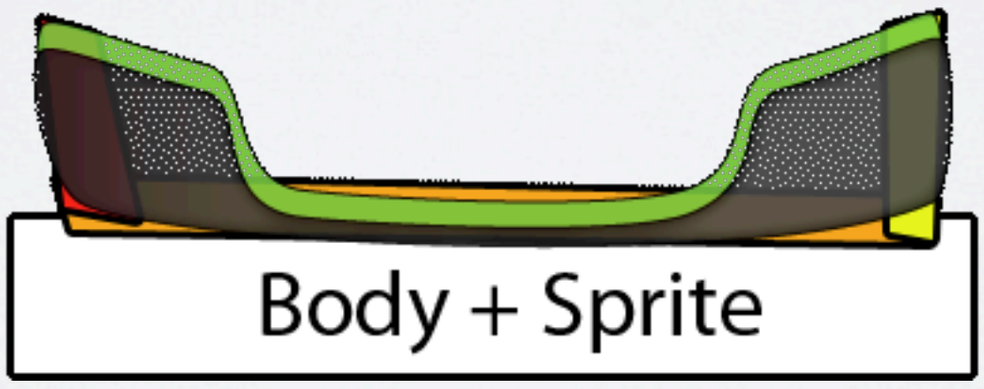
SECTION 4 - PART 2

- Decorating Chipmunk bodies with sprites
- Updating sprite position/angle based on sim
- Destroying sprites
- Demo: Decorating with Sprites!

DECORATING BODIES WITH SPRITES



=



DECORATING BODIES WITH SPRITES

- Subclass of CCSprite, stores associated Chipmunk body
- Method to be called each frame:

```
- (void)update {
    self.position = body->p;
    self.rotation =
     CC_RADIANS_TO_DEGREES(-1 * body->a);
}
```

- Consider storing backpointer to sprite with body->data

DESTROYING BODIES

```
cpSpaceRemoveBody(space, body);
cpSpaceRemoveShape(space, shape);
[self removeFromParentAndCleanup:YES];
```

- Cannot destroy a body/shape in a collision callback
- Workaround: post step callback

CODE+DEMO

Decorating with Sprites!

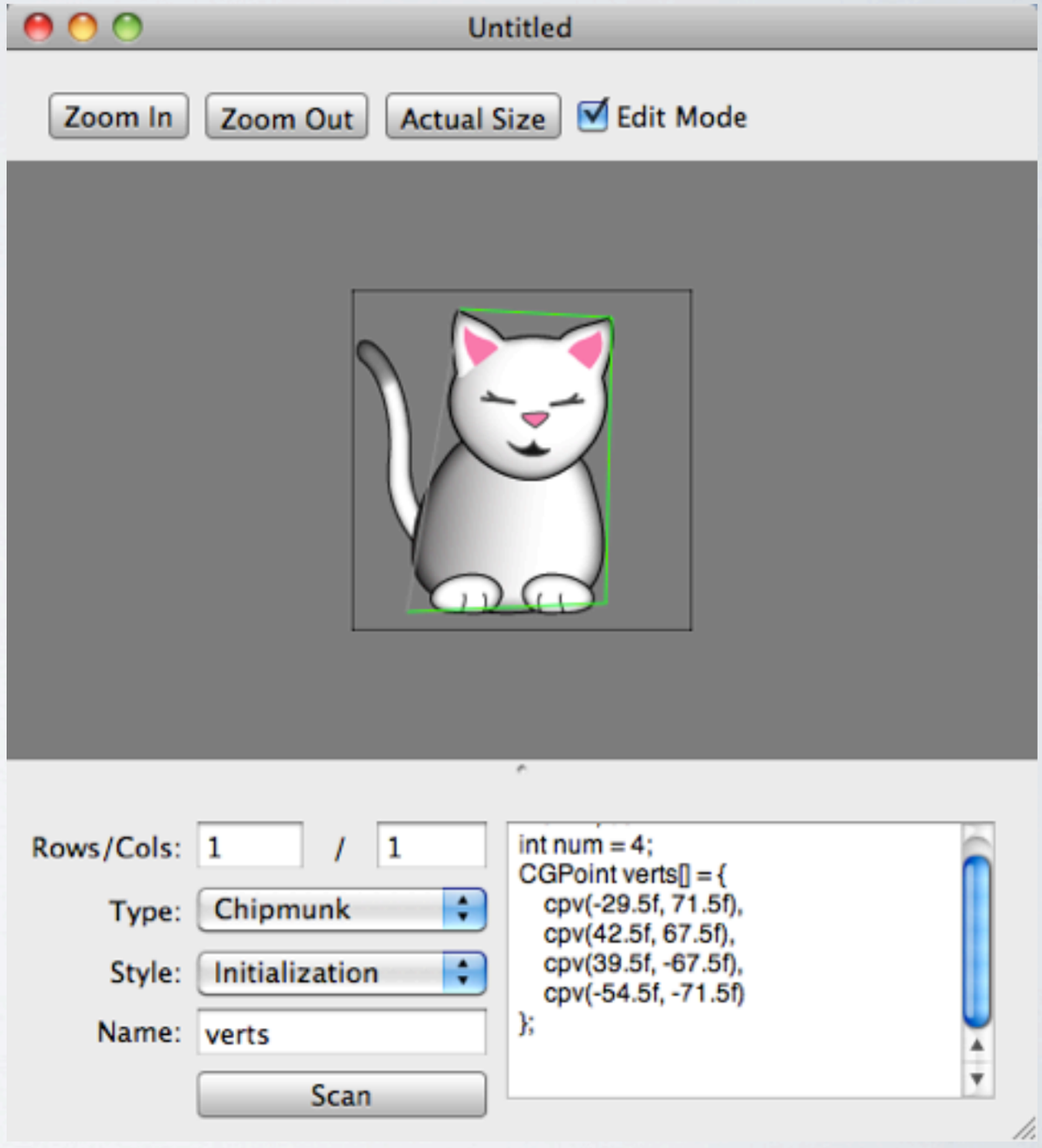
Cocos2DBook.com

SECTION 4 - PART 3

- Polygon shapes
- Demo: Cat -> polygon shape
- Multi-shape (and multi-sprite!) bodies
- Demo: Adding cat bed
- Collision detection
- Demo: Detecting win/loss conditions, and finishing touches

POLYGON SHAPES

- Need to specify vertex coordinates
- Use Vertex Helper to get
- Click to define verts
- Free version on github
- Pro version on App Store
- Also see Physics Editor



POLYGON SHAPES

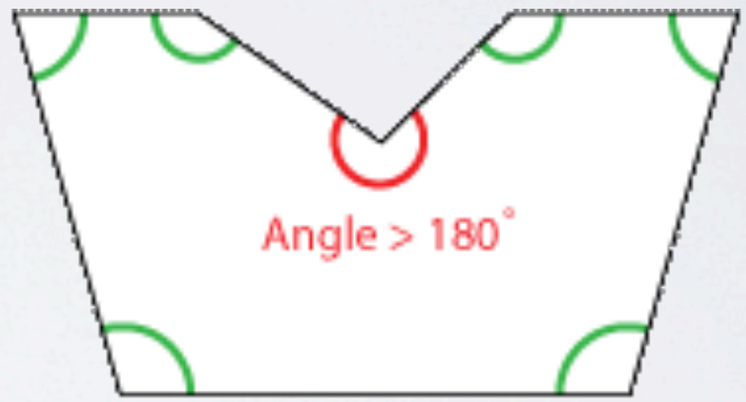
```
float mass = 1.0;
float moment = cpMomentForPoly(mass, num,
    verts, CGPointZero);
body = cpBodyNew(mass, moment);
body->p = location;
body->data = self;
cpSpaceAddBody(space, body);

shape = cpPolyShapeNew(body, num, verts,
    CGPointZero);
shape->e = 0.3;
shape->u = 1.0;
shape->data = self;
cpSpaceAddShape(space, shape);
```

POLYGON SHAPES GOTCHAS

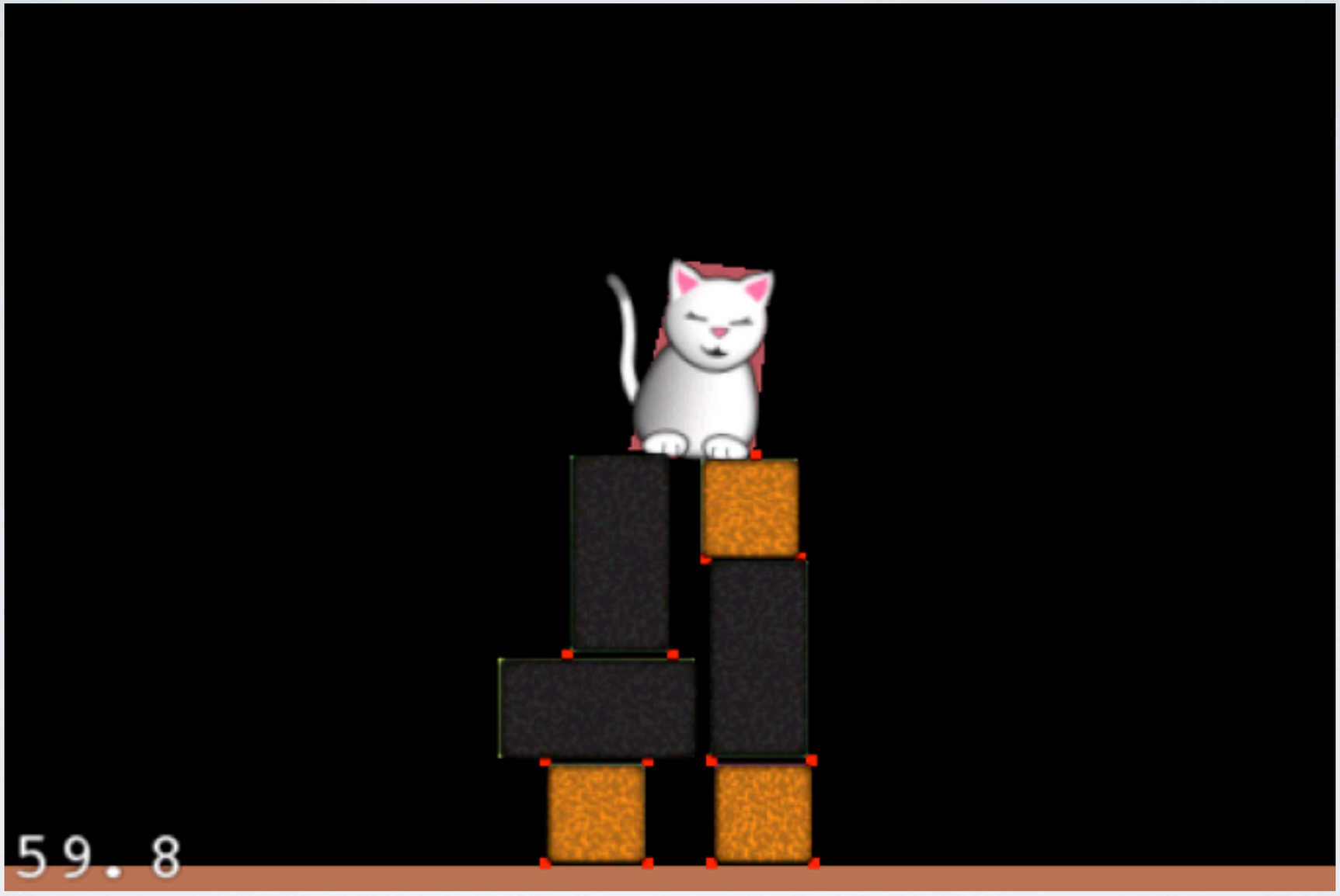
- Vertices must be defined in CW order (CCW for Box2D!)
- Polygons must be convex

Convex



Concave

- Need it to be concave? Use multiple shapes.

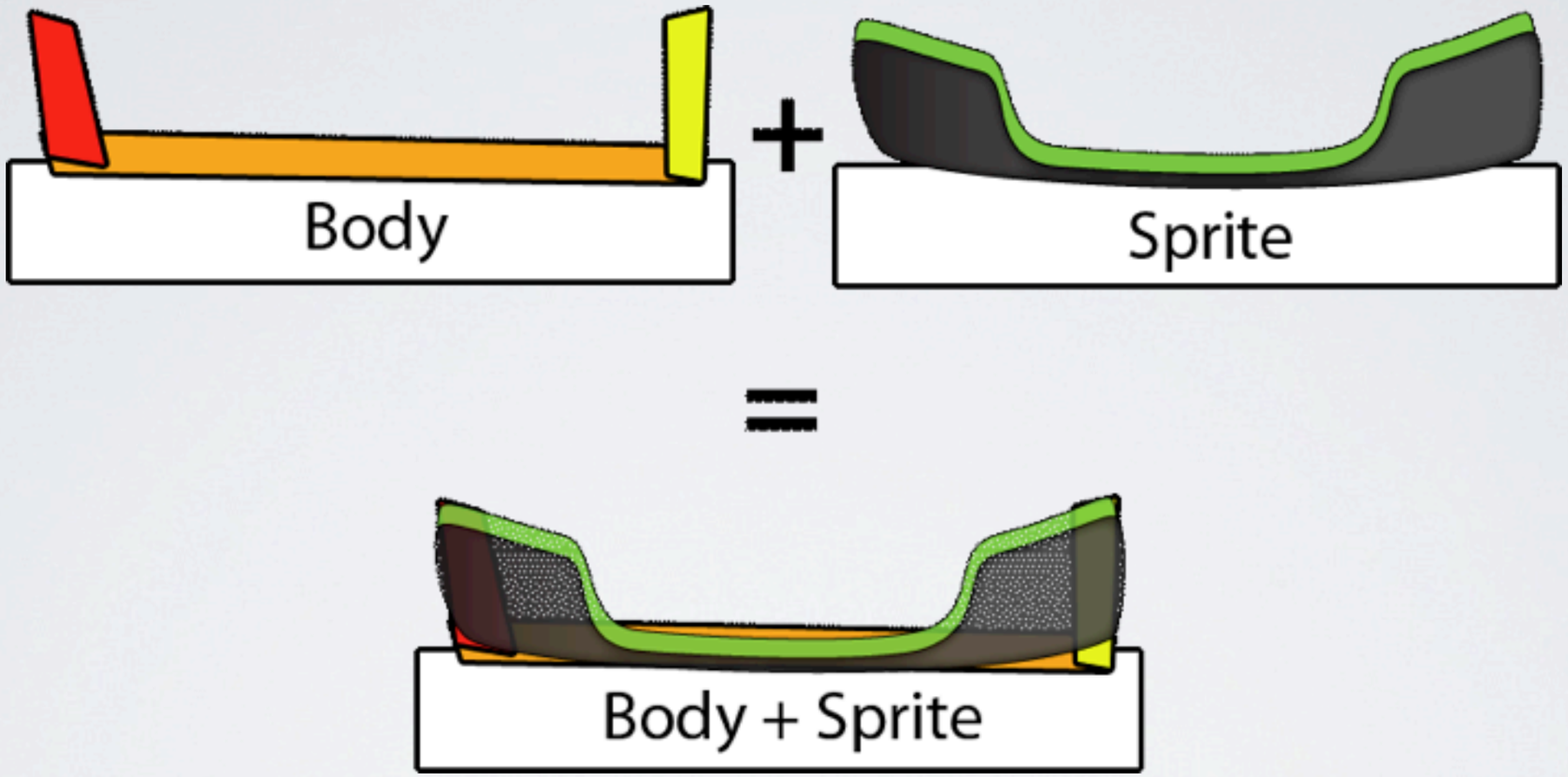


CODE+DEMO

Using a Polygon shape for the Cat

Cocos2DBook.com

MULTI-SHAPE BODIES



- Simply create multiple shapes on the same body
- Add mass & moments

MULTI-SPRITE BODIES

- One solution: if using the CPSprite method, associate both sprites to the same body!
  - Update method will put location of both sprites to body location
- Main "CatBed" subclass to create both

CODE+DEMO

Adding the Cat Bed

COLLISION DETECTION

- Register collision handlers

```
cpSpaceAddCollisionHandler(space,
 kCollisionTypeCat, kCollisionTypeBed,
 catHitBed, NULL, NULL, catSeparateBed,
 self);
```

- First parameters: space, collision types to check
- Then collision callbacks: begin, preSolve, postSolve, separate
- Last parameter: object to pass to collision callbacks

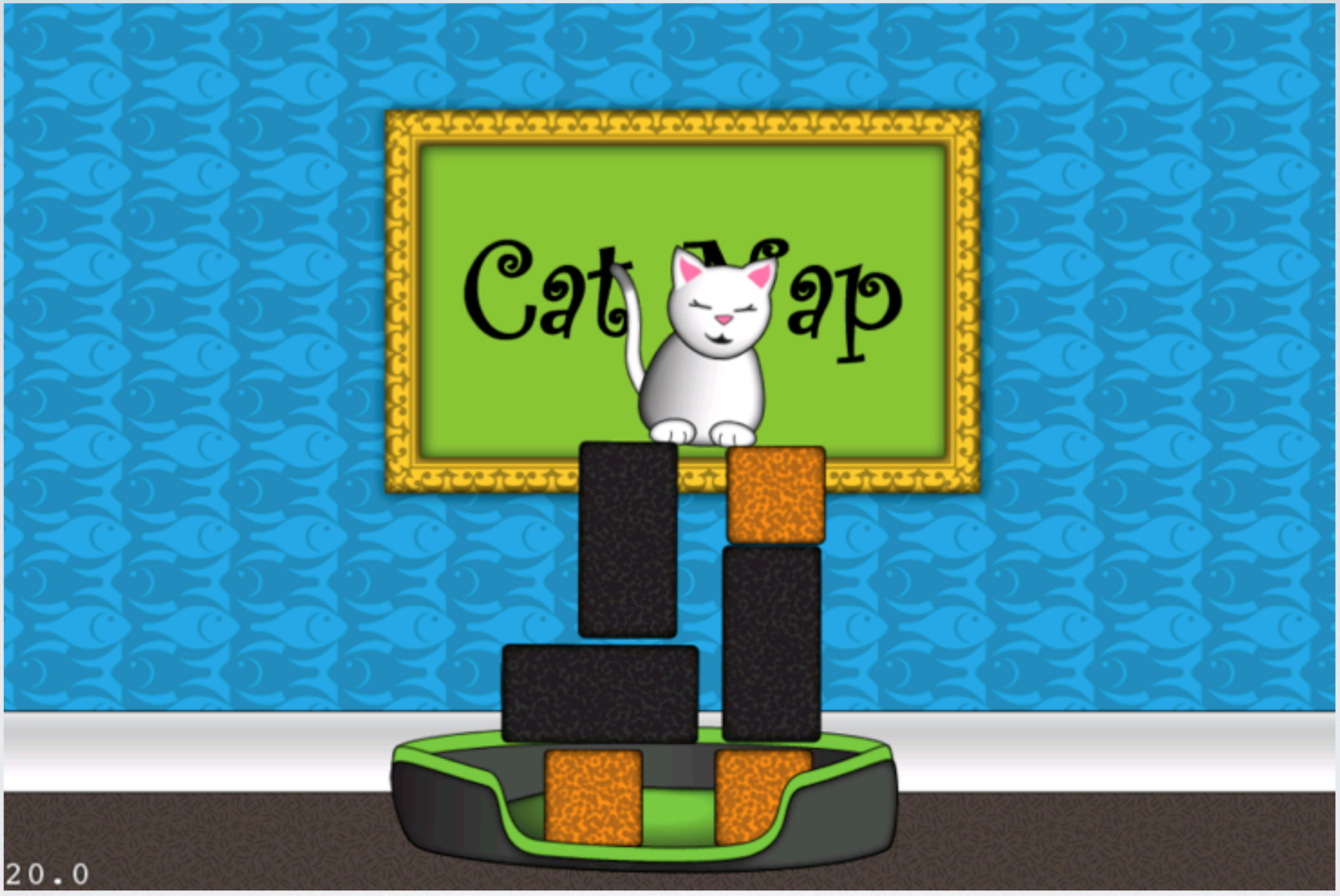
COLLISION DETECTION

- Callbacks are c-functions, generally nice to switch back to object:

```
cpBool catHitBed(cpArbiter *arb, struct
  cpSpace *space, void *data) {
    ActionLayer *layer = (ActionLayer *)data;
    [layer catHitBed];
    return cpTrue;
}
```

- Then do what you want!

```
- (void)catHitBed {
    CCLOG(@"Cat hit the bed!");
}
```



CODE+DEMO

Collision Detection, Final Game Walkthrough

SECTION RE-CAP

- Did you understand the steps to create a basic Chipmunk scene?
- Did you understand how to add your own bodies and shapes into the Chipmunk simulation?
- Did you understand how to synchronize the positions of Chipmunk bodies and Cocos2D sprites?
- Any questions?

READY FOR A CHALLENGE?

- Lab: Penguin Toss!
  - Create a basic Chipmunk scene
  - Add Penguin to game
  - Add ice blocks
  - Add ability to throw snowballs
  - Collision detection for the win!