テーマ1
enchant.js で ゲーム作り
(4)

2015/05/19

苫小牧高専 ソフトウェアテクノロジー部

今日やること

- テスト前の活動について
- ちょっとしたアンケート
- 前回のまとめ
- ゲームオーバーの処理を追加してみよう
- クラスを使ってみよう

前回のまとめ

インデント

どっちが見やすい？

インデントあり

```
enchant();

window.onload = function() {
    var core = new Core(320, 320);
    core.preload('chara1.png');

    core.onload = function() {
        var bear = new Sprite(32, 32);
        bear.image = core.assets['chara1.png'];
        bear.x = 0;
        bear.y = 0;

        core.rootScene.addChild(bear);
    }
    core.start();
}
```

インデントなし

```
enchant();

window.onload = function() {
var core = new Core(320, 320);
core.preload('chara1.png');

core.onload = function() {
var bear = new Sprite(32, 32);
bear.image = core.assets['chara1.png'];
bear.x = 0;
bear.y = 0;

core.rootScene.addChild(bear);
}
core.start();
}
```

インデントとは

- インデント ＝ 字下げ

- インデントを使って、
  入れ子になっている部分をわかりやすくするべし。

- インデントは、キーボードの[tab]キーで！
  （スペースキー連打はダメよ）

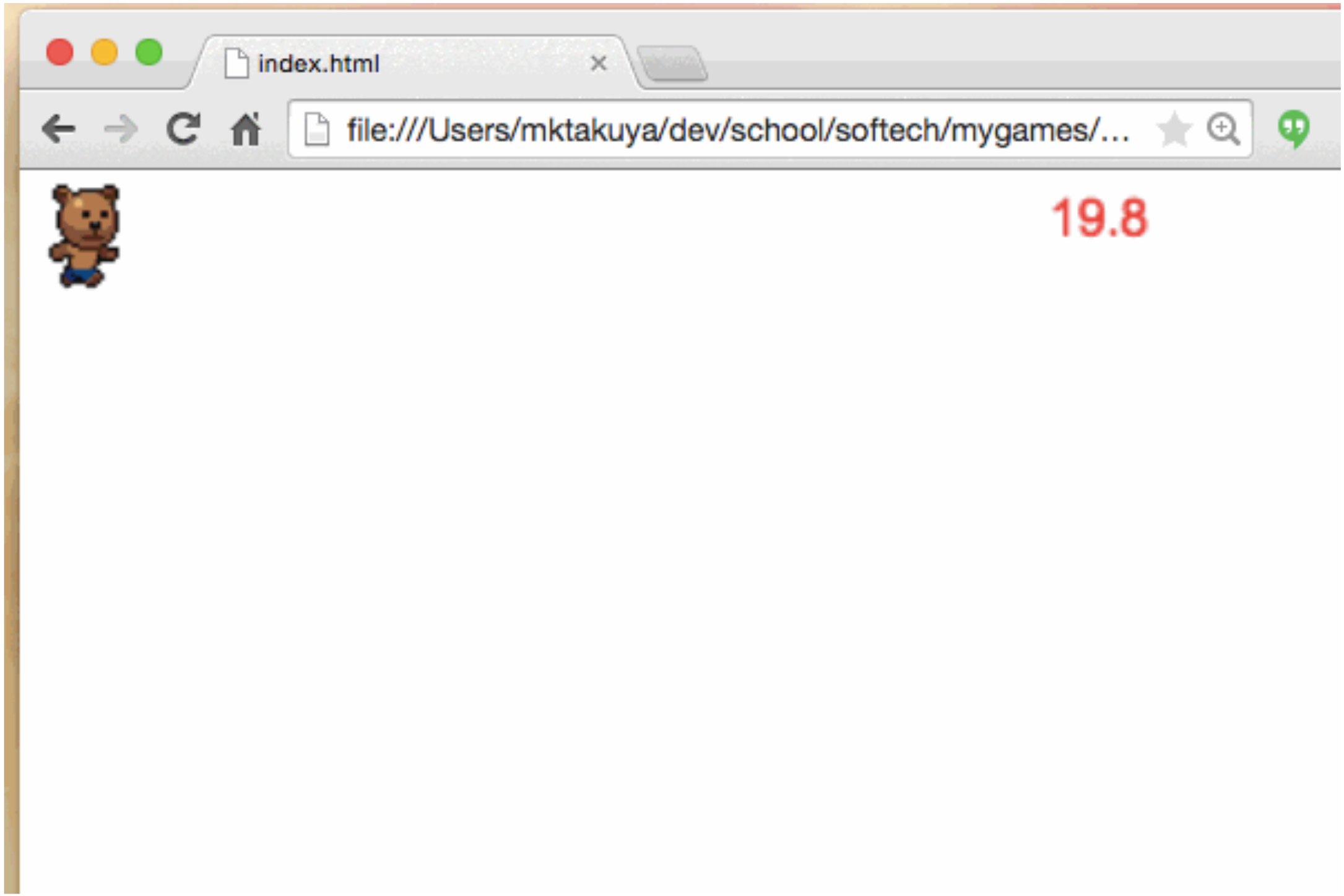
- 自分と、他人（3日後の自分も他人！）が
  見やすいコードを書くよう心がけよう！

前回のテーマ(1)

ラベルを表示してみる

～ストップウォッチを表示してみよう～

ストップウォッチ的な


index.html
file:///Users/mktakuya/dev/school/softech/mygames/…
19.8

main.js

```
        // 省略
        if (core.input.down) {
            this.y += 3;
        }
    });

    var label = new Label();← labelを作成
    label.x = 280;          ← 座標指定（おなじみ）
    label.y = 5;
    label.color = 'red';← 色指定
    label.font = '14px "Arial"';   ← フォントとサイズ
    label.text = '0';

    label.on('enterframe', function() {    ← たぶんaddEventListenerと同じ
        label.text = core.frame / core.fps;
    });

    core.rootScene.addChild(bear);
    core.rootScene.addChild(label);        ← addChildを忘れずに！
}
    core.start();
}
```

label.on

```
label.on('enterframe', function() {
    label.text = core.frame / core.fps;
});
```

- たぶんaddEventListenerと同じ

- label.text は labelに表示されるテキスト。
  そこに core.frame / core.fps を格納している。

- 現在のフレーム数をfpsで割れば秒数が出るよ！！

前回のテーマ(2)

衝突判定をやってみる

～いよいよゲームっぽくなってきたかも～

within

```
if (this.within(enemy, 15)) {
    label.text = 'HIT!';
}
```

- thisはbear

- this.within(enemy, 15)
  thisの中心とenemyの中心が15px近づいたらHIT!

- 数値を変えれば衝突判定を厳しくしたり、
  ゆるくしたりできる。

今日のテーマ(1)

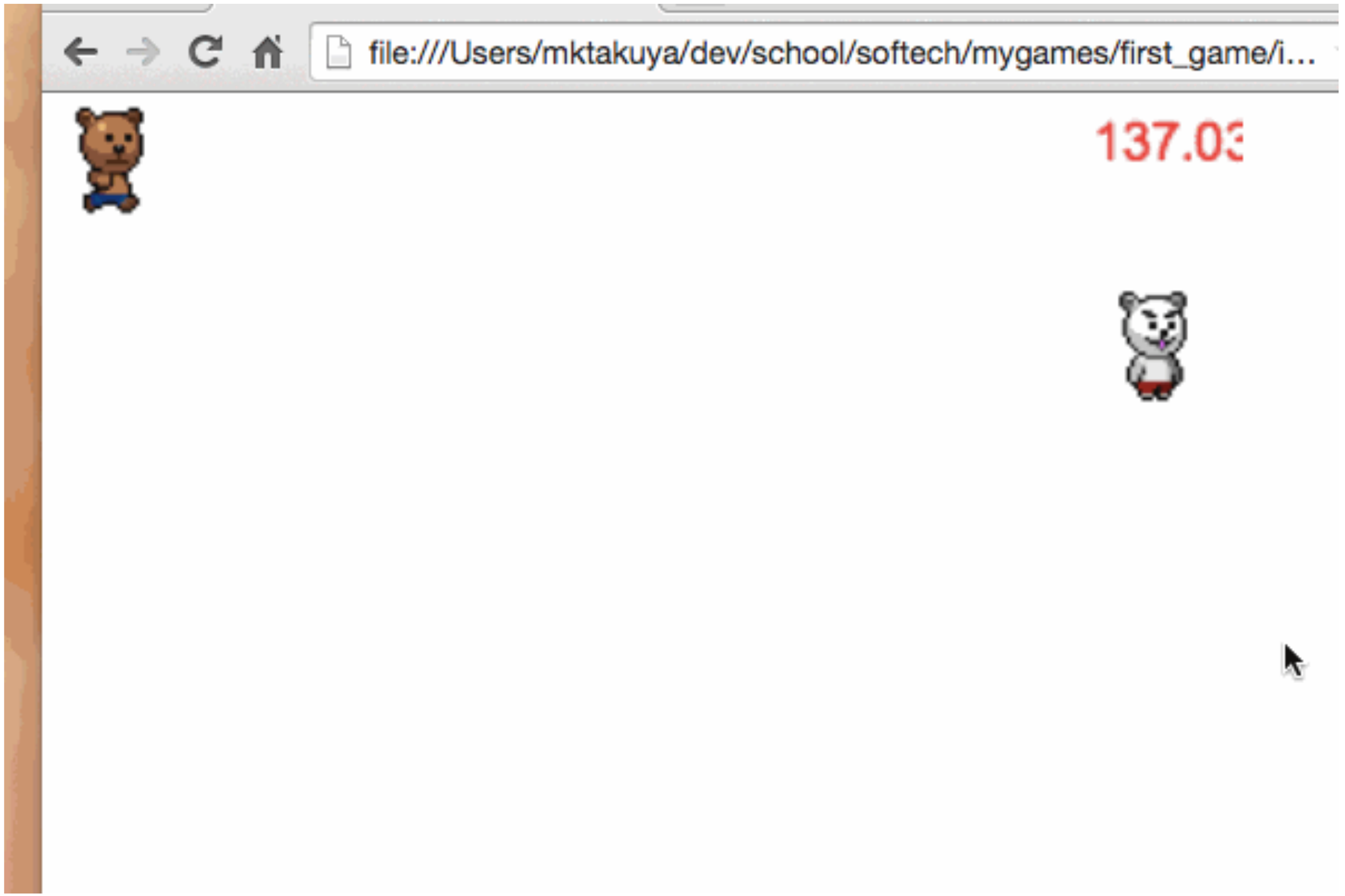
ゲームオーバーの処理

～いよいよゲームっぽくなってきたかも～

main.js

```
// 略
core.onload = function() {
    var bear = new Sprite(32, 32);

    // 中略

    var gameOverScene = new Scene();
    gameOverScene.backgroundColor = 'black';

    var gameOverLabel = new Label();
    gameOverLabel.x = 100;
    gameOverLabel.y = 160;
    gameOverLabel.color = 'red';
    gameOverLabel.font = '24px "Arial"';
    gameOverLabel.text = 'GAME OVER!!';

  bear.addEventListener('enterframe', function() {
   // 中略

    if (this.within(enemy, 15)) {
        core.pushScene(gameOverScene);
        gameOverScene.addChild(gameOverLabel);
        core.stop();
    }
  });

// 略
```

ゲームオーバー処理


file:///Users/mktakuya/dev/school/softech/mygames/first_game/i…
137.03

シーンについて復習

```
// 第1回目で書いたソースコード
enchant();

window.onload = function() {
    var core = new Core(320, 320);
    core.preload('chara1.png');

    core.onload = function() {
        // 中略
        core.rootScene.addChild(bear);
    }
    core.start();
}
```

- coreオブジェクトを作成。
  coreオブジェクトは最初からrootSceneを持っている。

- クマさん（Sprite）やテキスト（Label）を作るたびに、
  core.rootSceneにaddChildしていく。

自作のシーンを使う

- 自作のシーンをrootSceneに重ねるイメージ。

ゲームオーバー処理を作る

- rootSceneを自作シーン（gameOverScene）で覆い被さるイメージ。

- 自作シーンもSpriteやLabelと同様に作る。

```
var gameOverScene = new Scene();
gameOverScene.backgroundColor = 'black';
```

- シーンを出す処理

```
if (this.within(enemy, 15a)) { // 前回やった within
    core.pushScene(gameOverScene); // core.pushScene(yourSceneName)でシーンを出す
    core.stop();  // ゲームをstop
}
```

- rootSceneと同様にaddChildもできる

```
gameOverScene.addChild(gameOverLabel); // gameOverLabelは前のスライド参照
```

今日のテーマ(2)

クラスを使ってみよう

～いよいよゲームっぽくなってきたかも～

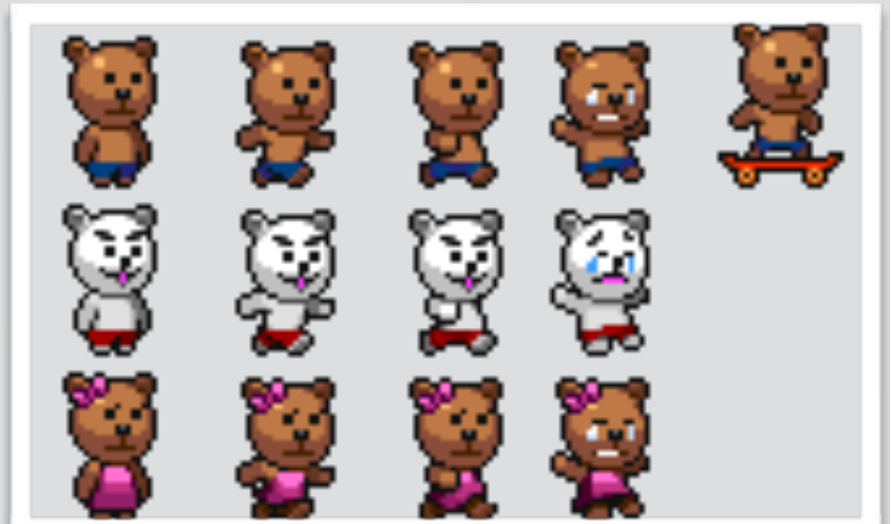
突然ですが、そろそろ暖かくなってきてクマも冬眠から覚める頃かと思うので、大量にクマさんを表示してみたいと思います。手始めに白いクマさん（Sprite）を50体作って画面に表示してください。

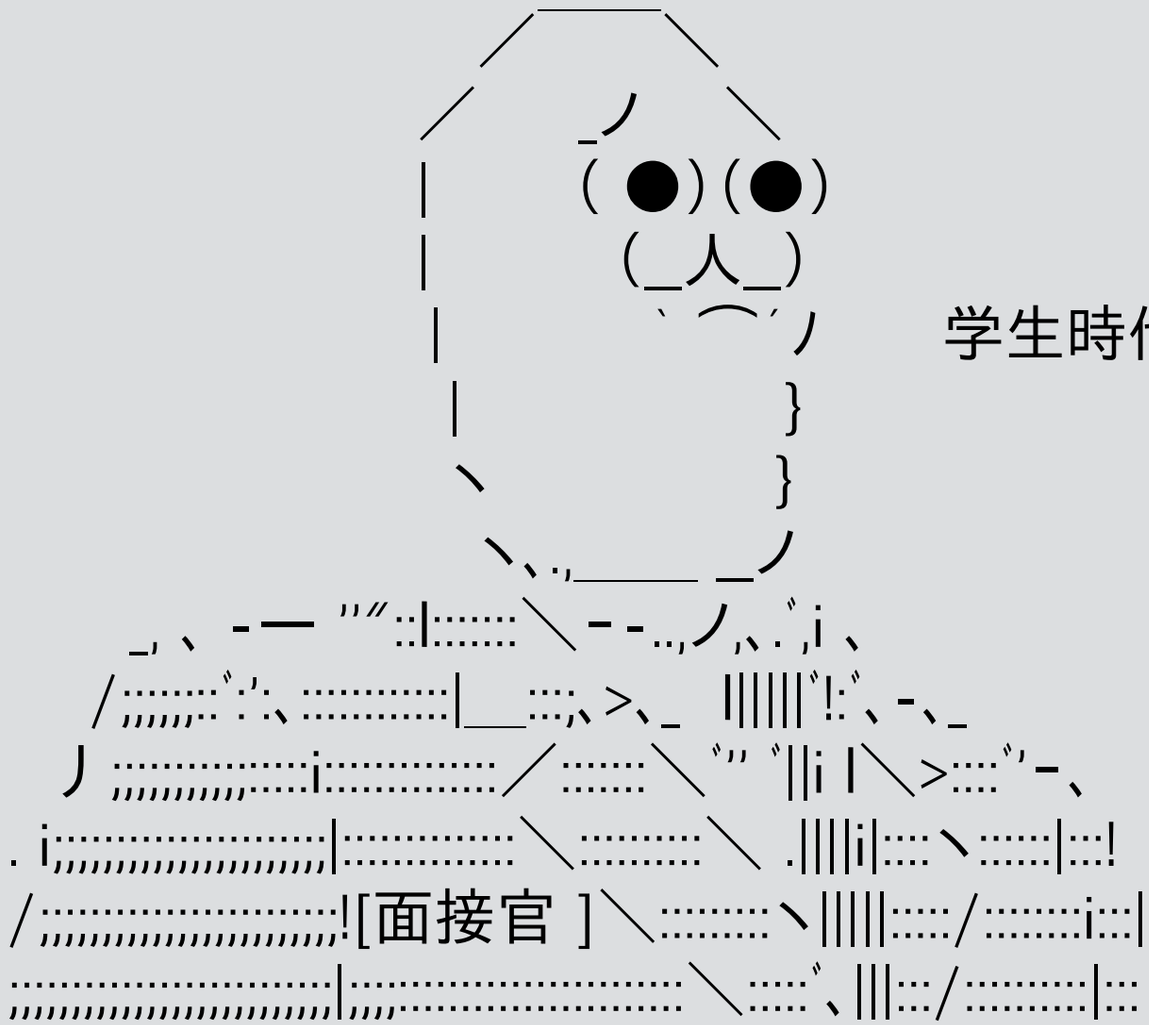
```
var bear1 = new Sprite(32, 32);
bear1.image = core.assets['chara1.png'];
bear1.x = 280;
bear1.y = 50;
bear1.frame = 5;

var bear2 = new Sprite(32, 32);
bear2.image = core.assets['chara1.png'];
bear2.x = 200;                          // クマさん同士が重ならないように適当に
bear2.y = 30;                           // 座標を変えてね
bear2.frame = 5;

var bear3 = new Sprite(32, 32);
bear3.image = core.assets['chara1.png'];
bear3.x = 100;
bear3.y = 50;
bear3.frame = 5;

var bear4 = new Sprite(32, 32);
bear4.image = core.assets[‘chara1.png'];
// 略
// これを40回以上繰り返す（頑張れ
```

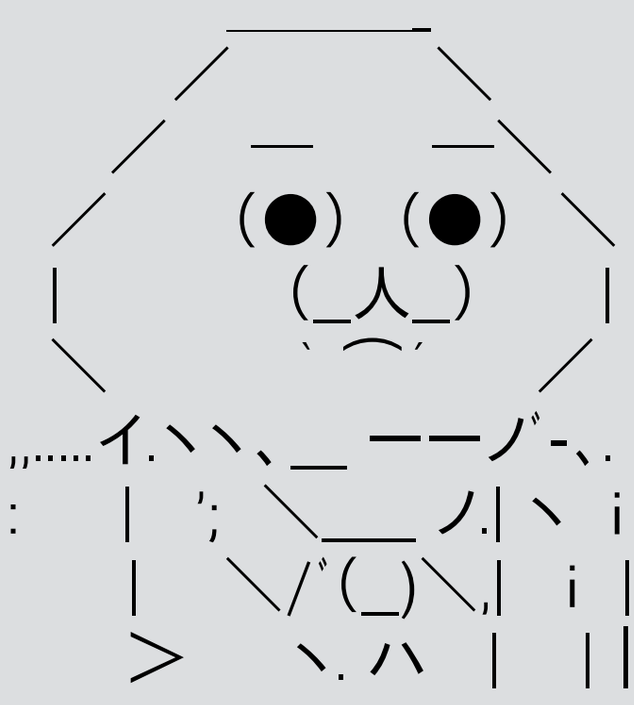
さて、50体の白いクマさんを作ってもらいましたが、やっぱり白いクマさんだとありきたりなので、白いクマさんじゃなくてピンクのクマさんに変更してください。

```
var bear1 = new Sprite(32, 32);
bear1.image = core.assets['chara1.png'];
bear1.x = 280;
bear1.y = 50;
bear1.frame = 5;

var bear2 = new Sprite(32, 32);
bear2.image = core.assets['chara1.png'];
bear2.x = 280;
bear2.y = 50;
bear2.frame = 5;

var bear3 = new Sprite(32, 32);
bear3.image = core.assets['chara1.png'];
bear3.x = 280;
bear3.y = 50;
bear3.frame = 5;

var bear4 = new Sprite(32, 32);
bear4.image = core.assets[‘chara1.png'];
// 略
```

← ここを bearN.frame = 9 にする × 50回




学生時代に打ち込んだものは？
面接官


キーボードです

```
var bear1 = new Sprite(32, 32);
bear1.image = core.assets['chara1.png'];
bear1.x = 280;
bear1.y = 50;
bear1.frame = 5;

var bear2 = new Sprite(32, 32);
bear2.image = core.assets['chara1.png'];
bear2.x = 280;
bear2.y = 50;
bear2.frame = 5;

var bear3 = new Sprite(32, 32);
bear3.image = core.assets['chara1.png'];
bear3.x = 280;
bear3.y = 50;
bear3.frame = 5;

var bear4 = new Sprite(32, 32);
bear4.image = core.assets[‘chara1.png'];
// 略
// これを40回以上繰り返す（頑張れ
```

1. bearN（Nは自然数）を作成
2. bear.imageの指定
3. bearの座標を指定
4. bearのframeを指定
5. rootScene.addChild

やってることは皆同じ。
違うのは、座標だけ。

共通部分は

まとめられるのでは？

クマさんの共通部分

- Sprite型の変数である
- this.image は chara1.png である
- 1フレームごとにx += 5する処理
- 画面の端へ行ったらx座標を0にする的な処理

など

```
enchant();

window.onload = function() {
    var core = new Core(320, 320);
    core.preload('chara1.png');

    core.onload = function() {
        var Bear = Class.create(Sprite, {
            initialize: function(x, y) {
                Sprite.call(this, 32, 32);
                this.x = x;
                this.y = y;
                this.image = core.assets['chara1.png'];

                this.addEventListener('enterframe', function() {
                    this.frame = this.age % 3;

                    this.x += 3;
                    if (this.x > 320) {
                        this.x = 0;
                    }
                });
                core.rootScene.addChild(this);
            }
        });

        var bear = new Bear(0, 0);
    }
    core.start();
}
```

せっかくなので大々的に書き直そう

一度Bearクラスを定義してしまえば…

```
var bear = new Bear(0, 0);
var bear2 = new Bear(10, 0);
var bear3 = new Bear(30, 20);
var bear4 = new Bear(50, 100);     // 変えるのは座標だけでOK! もっと楽にやる方法もあるヨ
```

クラス

クラス

- オブジェクトが持つデータと振る舞いを定義した設計図のようなもの。
  オブジェクト： クマさん（Sprite）とか
  ストップウォッチ（Label）とか

- 詳しい話は次回する（かも）

もうちょっとだけクラス

- もし、犬を表すクラス Dog があったら……

- dog = new Dog(40, 8) // 身長40cm 体重8kg
  dog.name = “ポチ” // 名前データは犬
  dog.run() // 犬を走らせる

定義済みのクラスから
新しく変数を作る

```
var Bear = Class.create(Sprite, {
    initialize: function(x, y) {
        Sprite.call(this, 32, 32);
        this.x = x;
        this.y = y;
        this.image = core.assets['chara1.png'];

        this.addEventListener('enterframe', function() {
            this.frame = this.age % 3;

            this.x += 3;
            if (this.x > 320) {
                this.x = 0;
            }
        });
        core.rootScene.addChild(this);
    }
});
```

← 基本形

↑ おまじない

initialize: function(与える値) {
    初期化処理;
}

↑this.○○ でデータにアクセスできる。

←addChildも
ここでできる

クラスから
新しいインスタンスを生成

```
var bear = new Bear(0, 0);
var bear2 = new Bear(10, 30);
var bear3 = new Bear(50, 10);
var bear4 = new Bear(200, 50);
```

- var bear = new Bear(50, 100);
  というように変数を作れば、めんどくさい処理（座標の初期設定とか）を全部クラス側でやってくれる。

- initialize: **function**(x, y) { … } のx, yは
  こちらで指定できる部分。

課題1

1. Bearクラスに、
   “キーボードの矢印キーで移動する処理”
   を書き加えよう！

2. 敵を表すEnemyクラスを作ってみよう！
   ・敵とわかりやすいよう白クマさんに。
   ・初期座標と移動方向、スピードはランダムに。
   ・プレイヤーのクマさんと衝突したら
   ゲームオーバーにしてゲームをstopさせよう。
   ・Math.floor(Math.random() * 300);
   で 0以上300以下のランダムな整数が取れる。