

enchant.jsでゲームを作ろう

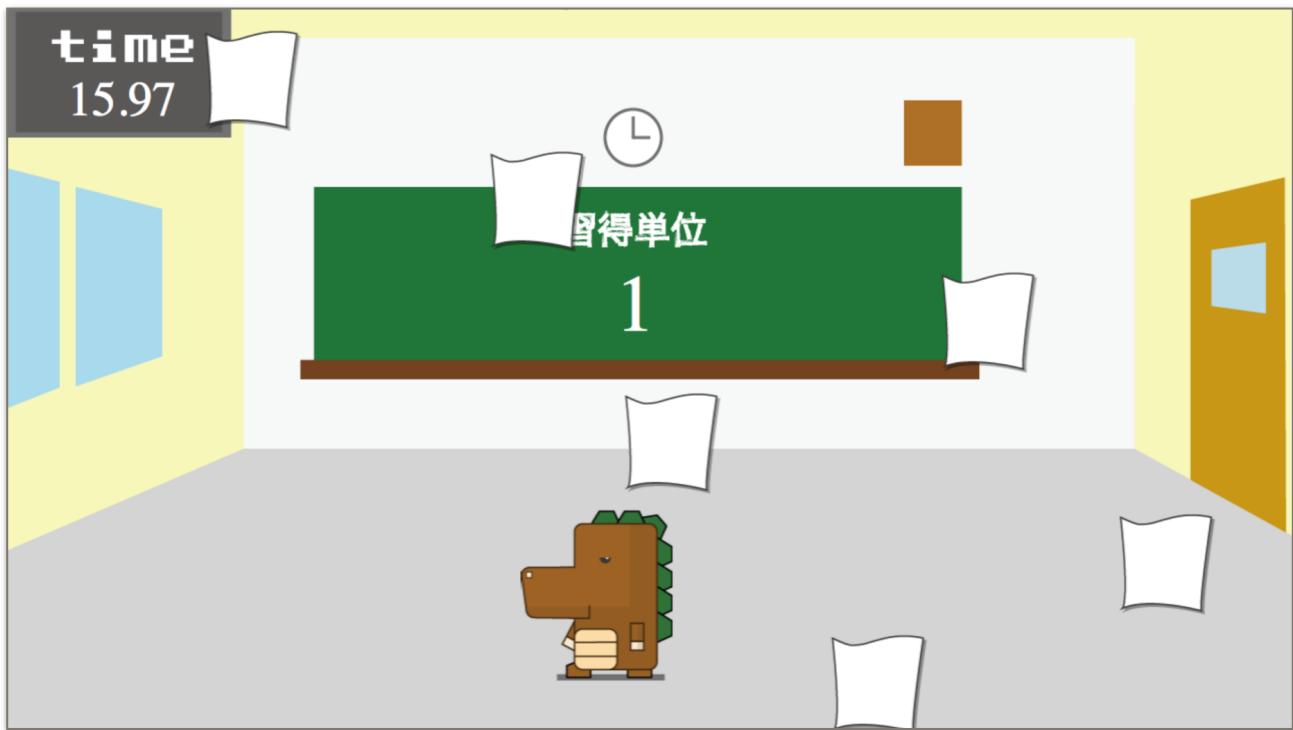
目次

序章 モルの単位キャッチ	3
第01章 ゲームの枠組みを作ろう	5
第02章 画像を読み込もう	7
第03章 ゲームシーンを作ろう	9
第04章 キャラクターを表示しよう	10
第05章 キャラクターを動かそう	12
第06章 キャラクターの向きを変えよう	14
第07章 背景を表示しよう	17
第08章 単位（プリント）を表示しよう	19
第09章 あたり判定をしてみよう	21
第10章 プリントを落下させよう	23
第11章 プリントをたくさん落とそう	25
第12章 ラベルを使って文字を表示しよう	27
第13章 ラベルを更新してスコアを表示しよう	29
第14章 キャラクターの動きを制限してゲーム性をつけよう	31
第15章 新しいゲームシーンを作ろう	31
第16章 新しいゲームシーンを表示してみよう	32
第17章 シーン移動をしよう	34

第18章 ラベルを使ってタイマーを作ろう	36
第19章 ラベルを毎フレーム更新して残り時間を表示しよう	37
第20章 ゲームオーバーシーンを作ろう	39
第21章 条件でシーン移動できるようにしよう	39
第22章 条件で背景画像を変えよう。	40
付録	42

序章 モルの単位キャッチ

今回作るゲームの説明



この本で作るゲームはモル（恐竜みたいな生物）が降ってくるプリントを獲得して単位を履修するゲームです。この本を通じてenchant.jsの基本概念と画像・文字の表示方法、あたり判定の仕方などを学んでいきます。

ファイル構成

モルの単位キャッチは9つのファイルで構成されています。それぞれ簡単に紹介しましょう。

1. *game.html*

enchant.jsはブラウザで動作するjavascript言語のライブラリです。今回のゲームはこのhtmlファイルをブラウザで開いて遊んでいきます。ブラウザで開いておきましょう。



2. *game.js*

このファイルにゲームの内容を書き込みます。このファイルをテキストエディタで開いておいてください。

game.html

3. *enchant.js*

enchant.jsでゲームを作るときに必要なファイルです。enchant.jsが持っている様々な機能はここに定義されています。



game.js

enchant.js

4. title.png

タイトル画面用の背景画像です。



5. game.png

ゲーム画面用の背景画像です。

title.png

game.png

6. game_over1.png

ゲームオーバー画面用の背景画像です。留年バージョン。



7. game_over2.png

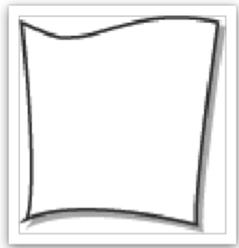
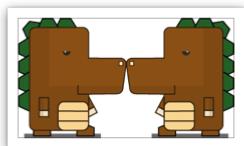
ゲームオーバー画面用の背景画像です。進級バージョン。

game_over1.png

game_over2.png

8. mol.png

主人公モルの画像です。



9. print.png

プリントの画像です。

mol.png

print.png

第01章 ゲームの枠組みを作ろう

[window.onload]coreの初期化

enchant.jsでゲームを作るためには、「enchant.jsライブラリを使うことを宣言する」「ゲームの入れ子を作る」この二つをまず始めにする必要があります。game.jsにコードを記述していきます。

```
GAMEN_YOKO = 1280;  
GAMEN_TATE = 720;  
  
enchant();  
  
var core = {};  
  
window.onload = function(){  
  
    core = new Core(GAMEN_YOKO, GAMEN_TATE);  
  
    core.onload = function(){  
    };  
  
    core.start();  
};
```

A. 画面の大きさを決める。

```
GAMEN_YOKO = 1280;  
GAMEN_TATE = 720;
```

まず画面の大きさを決めましょう。今回は横1280px、縦720pxとしました。この大きさはゲームの画面の大きさを決めるものですが、実際にはブラウザの大きさに対応してゲーム画面は拡大したり縮小したりします。設定を変えることでこの拡縮はやめられますが今回は変更しません。従ってこの大きさは、これから読み込む画像ファイルの基準となる程度に考えていただいて結構です。

B. enchantライブラリを読み込む

```
enchant();
```

上記のように記述することでグローバル空間でenchantライブラリの関数を使用できるようになります。

C. 入れ子を作る

```
window.onload = function(){

    core = new Core(GAMEN_YOKO, GAMEN_TATE);

    core.onload = function(){
    };

    core.start();
};
```

window.onloadはブラウザが用意している関数です。読み込み時に処理するコードを{}内に記述します。

新しい入れ子を作る為にCoreインスタンスを生成しています。new Core(引数1, 引数1)はenchantライブラリに定義されている関数で一つ目の引数を横幅、二つ目の引数を縦幅にしたゲームの入れ子を生成します。ここではnewすることによりcoreというCoreのインスタンスを生成しています。生成したCoreのインスタンスの中にキャラクターの画像や背景を表示するためのコードを書くことでゲームを作ります。

core.onload関数の中でゲームを作っていきます。window.onloadと似ていますがここでのonloadはCore関数のメソッドです。従ってwindow関数とは違ってstart()メソッドを呼びゲームの描画を開始する必要があります。

第02章 画像を読み込もう

[window.onload]gazouの読み込み

ゲームのキャラクターや背景はそれぞれ画像として保存されています。enchant.jsではゲームで使用する画像をゲーム開始前に読み込んでおく必要があります。今回はゲームに使用する画像をあらかじめすべて読み込んでおきましょう。

```
007
008 window.onload = function(){
009
010 core = new Core(GAMEN_YOKO, GAMEN_TATE);

011     var gazou = [];
012     gazou.push("title.png");
013     gazou.push("game.png");
014     gazou.push("game_over1.png");
015     gazou.push("game_over2.png");
016     gazou.push("mol.png");
017     gazou.push("print.png");

018     core.preload(gazou);

019
020     core.onload = function(){
021         core.start();
022     };
023
024 }
```

A. 画像を配列に格納する

```
var gazou = [];
gazou.push("title.png");
gazou.push("game.png");
gazou.push("game_over1.png");
gazou.push("game_over2.png");
gazou.push("mol.png");
gazou.push("print.png");
```

画像は配列<gazou>に格納してから読み込むようにします。以下のように読み込むことも可能ですがコードの可読性を考慮し、配列に格納してから読み込むようにしましょう。

```
core.preload("title.png", "game.png", "game_over1.png", "game_over2.png", "mol.png",
"print.png");
```

B. 画像を読み込む

```
core.preload(gazou);
```

preload()メソッドの言葉からわかるようにゲームが開始される前に画像は読み込む必要があります。

第03章 ゲームシーンを作ろう

[core.onload]ゲーム画面の生成と表示

ゲームを作り始めます。自分たちが普段しているゲームを思い出してください。ゲームにはいろいろな画面があると思います。タイトル画面、戦闘画面、メニュー画面、ゲームオーバー画面など、改めて考えてみるとゲームは様々な画面で構成されていることが分かりますね。enchant.jsではこれらの画面をシーン（Scene）と呼びます。そして、そのSceneに表示する画像や文字をScene（親）の子供としてひも付けることでゲームを構成しています。それではSceneを作成してみましょう。

```
019  
020 core.preload(gazou);  
021  
022 core.onload = function(){  
  
    var gameGamen = new Scene();  
    core.pushScene(gameGamen);  
  
023 };  
024  
025 core.start();  
026 };
```

A. Sceneを作成する

```
var gameGamen = new Scene();  
core.pushScene(gameGamen);
```

SceneクラスからgameGamenを生成しています。生成したgameGamenをcoreにcore.pushSceneすることで表示させます。

第04章 キャラクターを表示しよう

[GameGamen]モルの表示

ゲームの主人公であるモルを表示させてみましょう。enchant.jsでは画像を使用したオブジェクトをスプライト (Sprite) と呼びます。Spriteクラスから作成したインスタンスに画像を指定することで画像が表示できます。

```
023
024 var gameGamen = new Scene();
025 core.pushScene(gameGamen);
026
027     var mol = new Sprite(151, 169);
028     mol.image = core.assets["mol.png"];
029     mol.moveTo(300, 500);
030
031     gameGamen.addChild(mol);
032
033 };
034
035 core.start();
036 }
```

A. Spriteのインスタンスに画像を指定する

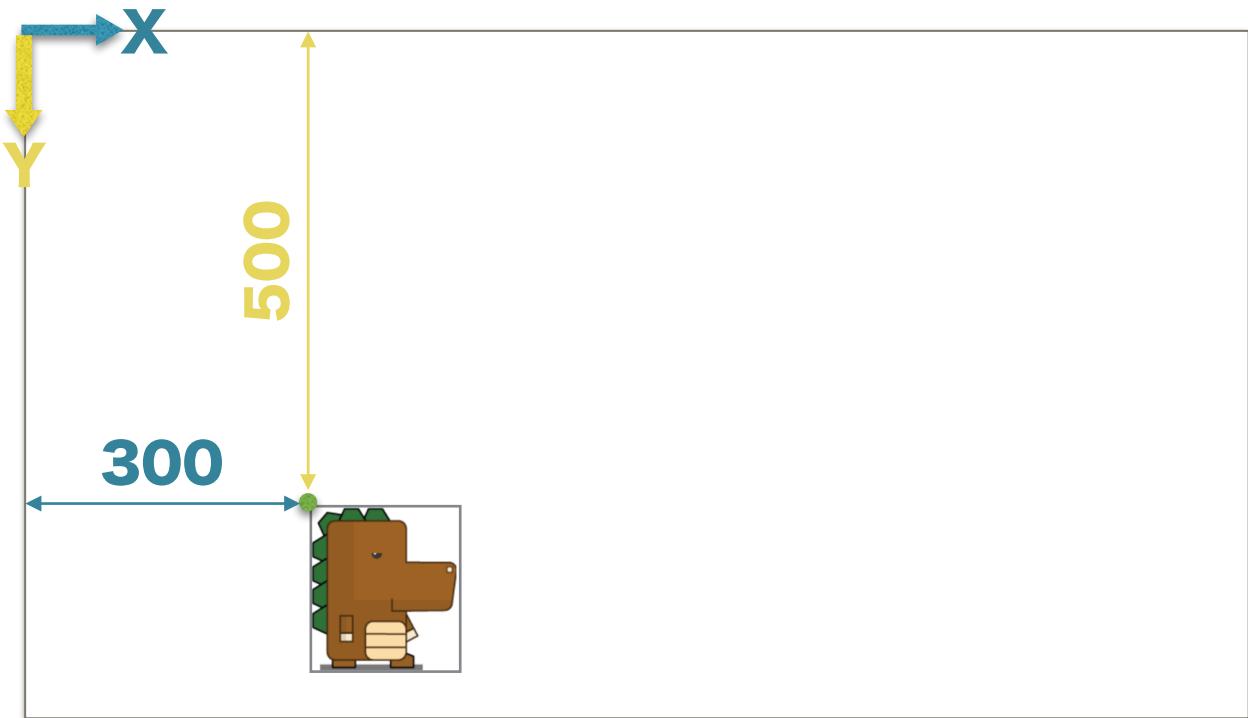
```
var mol = new Sprite(151, 169);
mol.image = core.assets["mol.png"];
```

Spriteクラスからmolを作成しましょう。Sprite()で指定している引数は表示したいもの大きさです。第一引数が横幅、第二引数が縦幅です。core.assets[]で第02章で読み込んだ画像を指定できます。mol.imageに代入しましょう。

B. 作成したスプライトの表示座標を変更する

```
mol.moveTo(300, 500);
```

moveToメソッドをしようしてmolを動かしましょう第一引数の300はx座標、第二引数の500はy座標を示しています。enchant.jsの座標系は以下のようになっているので注意が必要です。画像の左上の座標が指定されます。



C. Sceneに紐付ける

```
gameGamen.addChild(mol);
```

作成したmolをSceneに紐付けゲームに表示させましょう。addChild()メソッドを用いてSceneにmolを紐付けます。ここではSpriteであるmolを紐付けましたが他にも後述するLabelやGroupなど様々なものをSceneに紐づけることが可能です。

第05章 キャラクターを動かそう

[Mol]touchmoveに応じて動かす

enchant.jsでは「タッチした」や「タッチしながら動いている」などをイベントを発行することで感知できます。今回はtouchmove（タッチしながら動いてる）のイベントを監視してイベントが発行されたときにタッチした場所にモルを動かしています。前回指定したx=300とy=500は上書きされるので残りません。

```
027 var mol = new Sprite(151, 169);
028 mol.image = core.assets["mol.png"];
029 mol.moveTo(300, 500);
030
031 mol.addEventListener("touchmove", function(e){
032     this.x = e.x - this.width / 2;
033     this.y = e.y - this.height / 2;
034 });
035
036 gameGamen.addChild(mol);
037
038 };
```

A. Eventを受け取れるようにする

```
mol.addEventListener("touchmove", function(e){
});
```

addEventListener()メソッドでmolがイベントを受け取れるようにします。addEventListener()メソッドの引数は少しあわづらいますが二つあります。一つ目は"touchmove"ですね。第一引数に受け取りたいイベントを指定します。二つ目はfunction(e){}です。{}は改行されているのでわかりづらいかもしれませんがひと続きです。二つ目は関数をしているので予想できるかもしれません、イベントを受け取った後の処理を記述します。function(e){}が引数eを持っていることに注目してください。"touchmove"イベントを受け取ったときに実行される関数に引数を与えておくとenchant.jsがその引数にタッチした座標を代入してくれます。

B. タッチした座標からmolの座標を決める

```
this.x = e.x - this.width / 2;
this.y = e.y - this.height / 2;
```

thisという見たことのないものがありますね。これは英語の意味の通り「この」という意味です。では「この」とはコード上ではどこを指しているのでしょうか。このコードが書かれている場所に注目してください。mol.addEventListener()の中ですね。従ってここではmolのことを指していることになります。

先ほど説明したeが書かれていますね。touchmoveイベントを受け取ったときはe.xとe.yでタッチされたx座標、y座標が分かるということです。

C. ゲームを表示してみよう

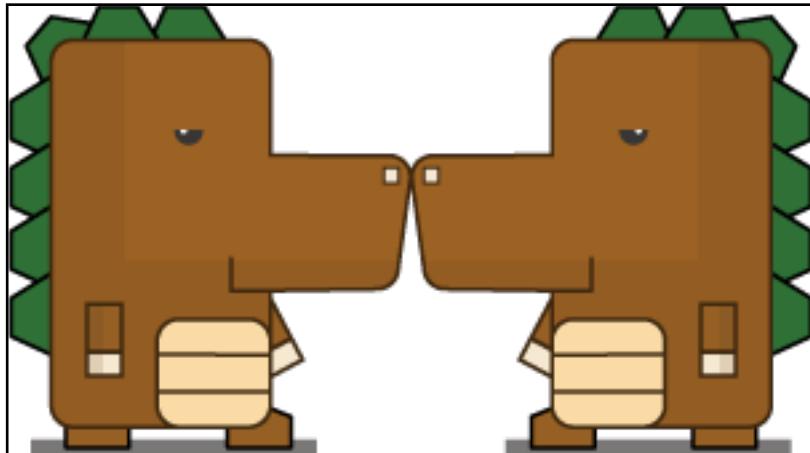
ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。以下のように表示されモルをドラッグできるはずです。



第06章 キャラクターの向きを変えよう

[Mol]touchmoveに応じてframeを変える

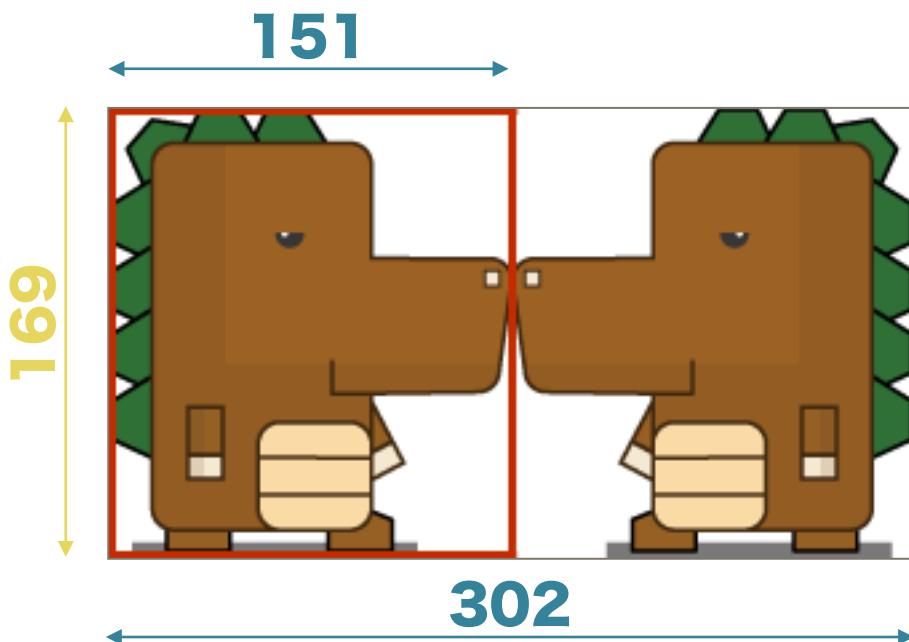
キャラクターが常に一方向を向いているのは面白くありません。動く方向によって向きを変えてみましょう。第04章でenchant.jsには画像を表示させるSpriteオブジェクトがあることが確認できましたね。実はこのSpriteには、指定した画像を変えずにSpriteに表示する絵柄を変えるframeという機能があります。まずフォルダに入っているmol.pngを見てみましょう。



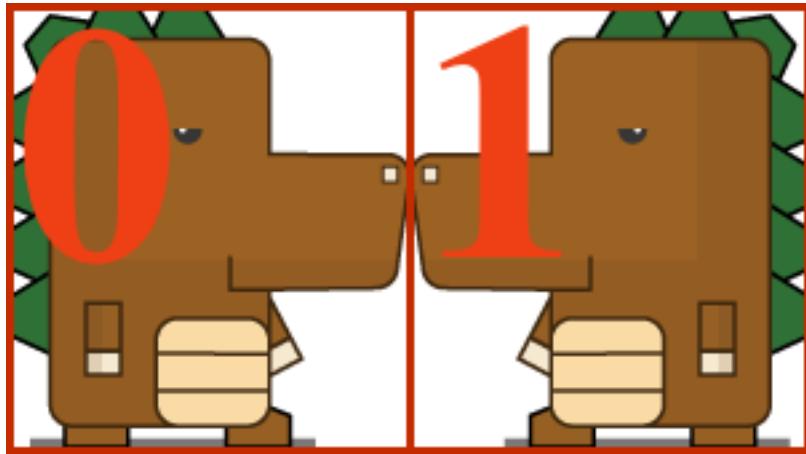
この画像がゲーム画面に表示されているモルと異なっているのがお分かりいただけると思います。ゲーム画面では一人しかいないモルがこの画像には二人いますね。ではなぜモルを一人だけ表示させることができたのか。その答えはSpriteの呼び出し方にあります。

```
027 var mol = new Sprite(151, 169);
```

151と169が表示したいものの横幅と縦幅を示すことは第04章で説明しました。この151と169は実際の画像では以下の赤い枠線の範囲を示しています。赤い枠に収まらない範囲はenchant.js



がSpriteに与えたframeという値を変えることで表示できます。Sprite()が呼ばれ画像が指定されたときenchant.jsは画像に対してSprite()で指定された引数の大きさで画像をタイル上に分割し、分割した領域に番号をつけていきます。従って今回の画像は以下のように番号が振られます。



```
028 mol.image = core.assets["mol.png"];
029 mol.moveTo(300, 500);
030
031 mol.addEventListener("touchmove", function(e){
    if(e.x - this.width / 2 > this.x){
        this.frame = 0;
    }else{
        this.frame = 1;
    }
032
033     this.x = e.x - this.width / 2;
034     this.y = e.y - this.height / 2;
035 });

});
```

A. タッチした座標で条件分岐する

```
if(e.x - this.width / 2 > this.x){

}else{

}
```

タッチした座標がモルの座標よりも右ならtrue、左ならfalseを返す条件式を用います。

B. フレームを用いて画像を入れ替える

```
if(e.x - this.width / 2 > this.x){
    this.frame = 0;
}else{
    this.frame = 1;
}
```

thisはmolのtouchmoveイベントの中の処理なのでmolのことを指しますね。 molのframeに条件式がtureなら0（右向き）、1（左向き）を代入しています。先ほどの画像と対応して考えてみてください。

C. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。以下のように表示されモルが動く方向によって向きを変えるはずです。



第07章 背景を表示しよう

[GameGamen]背景の表示

いままではゲームにとって大事なものが抜けていました。背景ですね。enchant.jsは背景も簡単に表示できます。というのも、enchant.jsでは背景もキャラクターも同じ画像オブジェクトとしてSpriteを使用して表示させているからです。つまり、この章で書くコードは第04章でモルを表示させる時に書いたコードとほとんど同じです。

```
023  
024 var gameGamen = new Scene();  
025 core.pushScene(gameGamen);  
026  
var gameHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);  
gameHaikei.image = core.assets["game.png"];  
gameGamen.addChild(gameHaikei);  
027  
028 var mol = new Sprite(151, 169);  
029 mol.image = core.assets["mol.png"];  
030 mol.moveTo(300, 500);
```

A. ゲーム画面の背景を表示する

```
var gameHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);  
gameHaikei.image = core.assets["game.png"];  
gameGamen.addChild(gameHaikei);
```

背景は画面サイズと同じにしたいので定数のGAMEN_YOKOとGAMEN_TATEを用いて画面と同じ大きさのSpriteを作成しましょう。今回、第04章と違う点はmoveTo()メソッドが呼ばれていないことでしょう。enchant.jsではmoveTo()メソッドを呼んでSpriteを移動させない場合座標(0, 0)にSpriteを表示します。背景は画面いっぱいに表示したいのでmoveTo()で移動させないまま表示しました。

B. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。背景が追加されると一気にゲームっぽくなりますね。



第08章 単位（プリント）を表示しよう

[Tani]initialize(表示)

この章からはモルがキャッチするプリントの作成に入りたいと思います。具体的にはprint.pngの表示です。

第08章は第04章、前章と同じSpriteの生成、表示です。今回は特に第04章と行っていることが全く同じなので余裕のある方は下記のコードを見ず、第04章コードを参考に書いてみてください。

```
044  });
045
046  gameGamen.addChild(mol);
047
var tani = new Sprite(95, 100);
tani.image = core.assets["print.png"];
tani.moveTo(400, 200);

gameGamen.addChild(tani);

048 }
049
050 core.start();
051 }
```

A. プリントを表示する

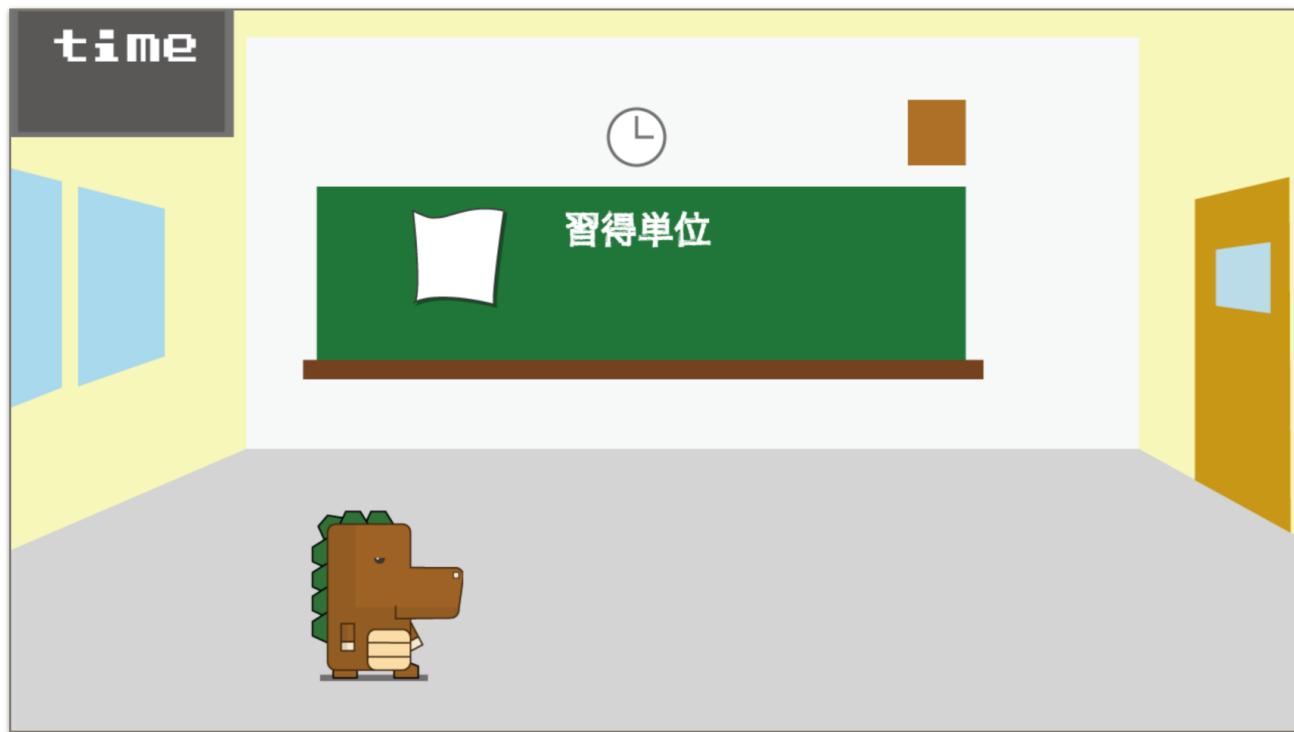
```
var tani = new Sprite(95, 100);
tani.image = core.assets["print.png"];
tani.moveTo(400, 200);

gameGamen.addChild(tani);
```

モルを作成したときと同じなので第04章を参考にしてください。

B. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。黒板の上にプリントが表示されたら成功です。

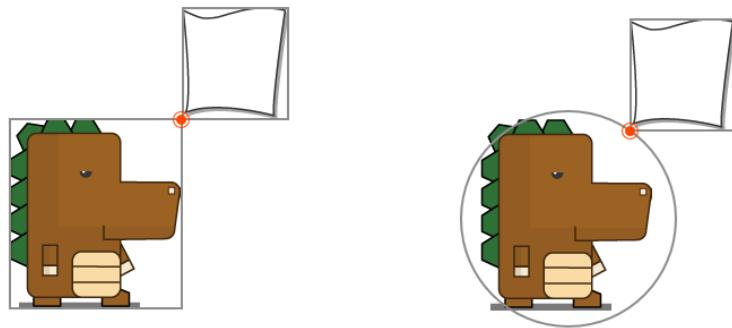


第09章 あたり判定をしてみよう

[Tani]プリントの衝突判定eventListener追加

モルがプリントに触れたらプリントが消えるようにしてみましょう。enchant.jsには二つのあたり判定方法があります。矩形判定のintersect(相手)メソッドと円形判定のwithin(相手, 距離)メソッドです。今回はモルとプリントを矩形判定してみましょう。

mol.intersect(print) mol.within(print, 80)



上記の図ではモルに大してあたり判定をしています。あたり判定はあたる側とあたられる側の二つのオブジェクトがありますが、それは入れ替え可能なのでモルに対してあたり判定をする場合とプリントに対してあたり判定をする場合の二通りあることが分かると思います。

```
050 tani.moveTo(400, 200);
051
052 gameGamen.addChild(tani);
053
054 tani.addEventListener("enterframe", function(){
  if(this.intersect(mol)){
    gameGamen.removeChild(this);
  }
});
055
056 core.start();
057 };
```

A. プリントに対してあたり判定を追加する

```
tani.addEventListener("enterframe", function(){
  if(this.intersect(mol)){
  }
});
```

あたっているかいないかの判定は毎フレーム行いたいのでenterframeイベントを受け取って実行するようにします。intersect()メソッドもwithin()メソッドもあたりの真偽を返すだけなのでそれをif条件式で掬ってやる必要があります。

B. あたったときの処理を書く

```
gameGamen.removeChild(this);
```

あたったときはtaniをゲーム画面から取り除きましょう。removeChild()でSceneからオブジェクトを取り除くという、addChild()と逆の動作ができます。

C. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。モルを動かしてプリントに触れさせ、プリントが消えたら成功です。



第10章 プリントを落下させよう

[Tani]等速の落下移動の実装

動かないプリントを獲得しても面白くないのでプリントを落下させてみましょう。この章ですることは三つです。

1. プリントの初期位置を画面の上端に設定する
 2. 每フレームプリントを動かす
 3. プリントが画面の外に出たら消す

それでは書いてみましょう

それでは冒頭のまじょう

```
046     gameGamen.addChild(mol);
047
048     var tani = new Sprite(95, 100);
049     tani.image = core.assets["print.png"];
050     tani.moveTo(400, 200);
051     tani.moveTo(400, -tani.height);
052
053
054     tani.addEventListener("enterframe", function(){
055         if(this.intersect(mol)){
056             gameGamen.removeChild(this);
057         }
058         this.y += 4;
059         if(this.y > GAMEN_TATE){
060             gameGamen.removeChild(this);
061         }
062     });
063
064 }
```

A. プリントの初期位置を画面の上端に設定する

```
tani.moveTo(400, -tani.height);
```

プリントのy座標を-tani.heightと自分自身の負の高さにすることで画面の上端にプリントの下端をあわせることができます。

B. 每フレームプリントを動かす

```
this.y += 4;
```

enterframeイベントを受けとるとたびにプリントのy座標に+4していくことでプリントが毎秒120px（1秒=30フレーム）落下していくようになりますね。

C. プリントが画面の外に出たら消す

```
if(this.y > GAMEN_TATE){  
    gameGamen.removeChild(this);  
}
```

忘れがちなのがこの処理です。この処理はプリントが多くなったときに役に立ちます。画面外に出たプリントを消さないでおくと繰り返し生成されるプリントによりゲームの処理が重くなるかもしれません。ですから、画面外に出たオブジェクトは消すようにした方が賢明です。

D. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。プリントが上から落ちてきたら成功です。



第11章 プリントをたくさん落とそう

[GameGamen]単位の発生(複数降らす)

プリントをたくさん落としてみましょう。プリントをたくさん落とすためには一定間隔でプリントを作成する必要があります。またプリントの出現位置のx座標をランダムにする必要があります。

```
044  });
045
046  gameGamen.addChild(mol);
047
048  var tani = new Sprite(95, 100);
049  tani.image = core.assets["print.png"];
050  tani.moveTo(400, -tani.height);
051
052  gameGamen.addChild(tani);
053
054  tani.addEventListener("enterframe", function(){
055    if(this.intersect(mol)){
056      gameGamen.removeChild(this);
057    }
058    this.y += 4;
059    if(this.y > GAMEN_TATE){
060      gameGamen.removeChild(this);
061    }
062  });
063
064  gameGamen.addEventListener("enterframe", function(){
065    if(this.age % 30 == 0){
066      var tani = new Sprite(95, 100);
067      tani.image = core.assets["print.png"];
068      tani.moveTo(Math.random() * (GAMEN_YOKO - tani.width), -tani.height);
069
070      gameGamen.addChild(tani);
071
072      tani.addEventListener("enterframe", function(){
073        if(this.intersect(mol)){
074          gameGamen.removeChild(this);
075        }
076        this.y += 4;
077        if(this.y > GAMEN_TATE){
078          gameGamen.removeChild(this);
079        }
080      });
081    }
082  });
083
084  core.start();
```

A. プリントの出現位置のx座標をランダムにする

```
tani.moveTo(Math.random() * (GAMEN_YOKO - tani.width), -tani.height);
```

前章ではy座標を変更しましたが今回はx座標を変更します。Math.random()関数は0～1の間の少數を返すjavascriptの関数です。これに画面の幅からプリントの幅を引いた値を掛けすることでプリントが画面のx座標のどこからでも出現できるようになります。

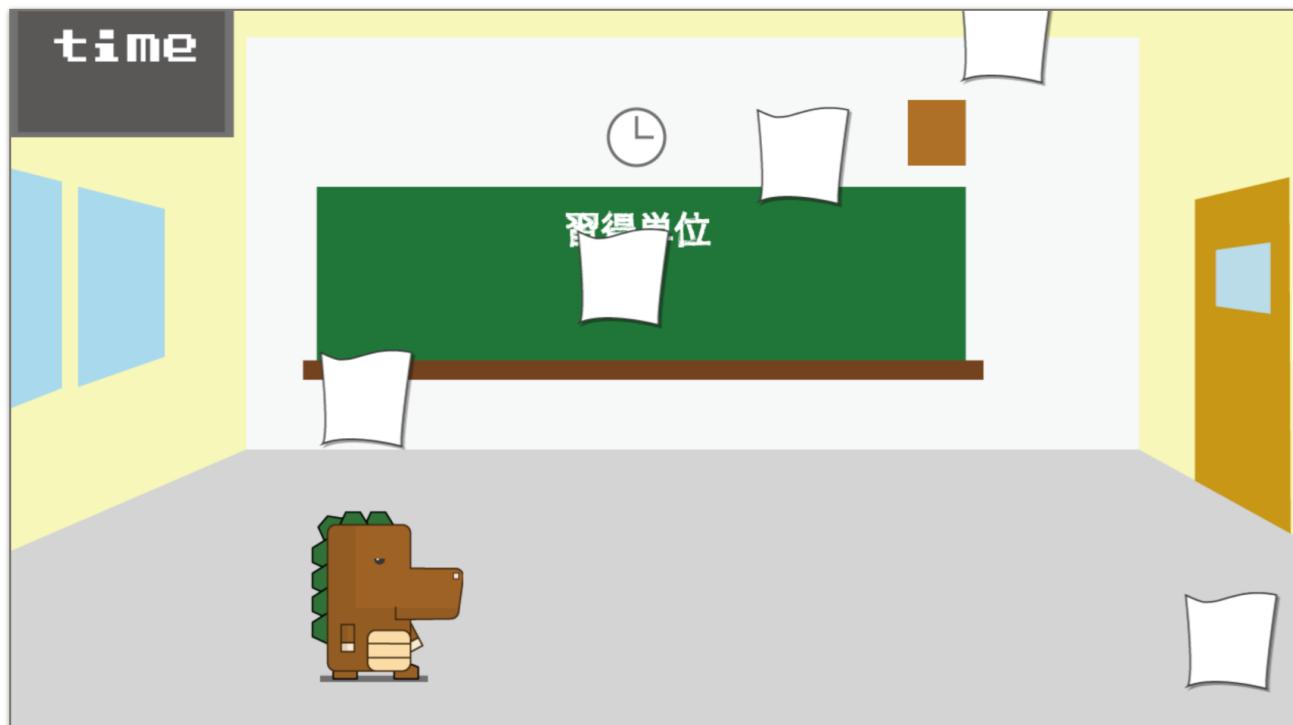
B. 一定間隔でプリントを作成する

```
gameGamen.addEventListener("enterframe", function(){
  if(this.age % 30 == 0){
  });
});
```

一定間隔で実行されるイベントと言えば毎フレーム実行されるenterframeイベントですね。しかし、毎フレームプリントを作成してしまうとさすがにプリントが多くなり過ぎです。30フレームに一回作成することにしましょう。上記にはthis.ageという値が出てきます。これはgameGamenが作成されてから何フレームたったのかを示す値です。毎フレームごとにこの値を30で割り、そのあまりが0であるときのみ真を返す条件式を使うことで30フレームに一回実行される関数が実装できます。それが上記のコードです。この中に第08～10章で書いたプリントのコードを移植します。そうすることで30フレーム（一秒間）に一回プリントが作成されるコードが実装できます。

D. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。



第12章 ラベルを使って文字を表示しよう

[TaniHyouji]単位数をつくる

Labelクラスを用いてゲーム画面に文字を表示しましょう。Labelクラスには特殊な操作はありませんが、たくさんの値があります。今回用いる値をまとめましょう。

text	Labelに表示する文字を指定します
width	Labelの幅を指定します
textAlign	Labelに表示する文字列の位置をleft,center,rightから指定します
font	文字の大きさとフォントファミリーを指定します
color	文字の色を指定します。

```
064      });
065  }
066  });
067
var taniHyouji = new Label();
taniHyouji.tani = 0;
taniHyouji.text = taniHyouji.tani;
taniHyouji.width = 600;
taniHyouji.textAlign = "center";
taniHyouji.font = "80px Serif";
taniHyouji.color = "white";
taniHyouji.moveTo(325, 250);

gameGamen.addChild(taniHyouji);

068  };
069
070 core.start();
071 }
```

A. 文字を表示する

```
var taniHyouji = new Label();
taniHyouji.tani = 0;
```

LabelクラスからtaniHyoujiインスタンスをつくりましょう。taniHyoujiにはtaniという値を作りあらかじめ0を代入しておきます。これをスコアと呼びましょう。プリントを獲得したらこの値を増やすようにします。

B. 文字の装飾をする

```
taniHyouji.text = taniHyouji.tani;
```

```
taniHyouji.width = 600;  
taniHyouji.textAlign = "center";  
taniHyouji.font = "80px Serif";  
taniHyouji.color = "white";  
taniHyouji.moveTo(325, 250);  
  
gameGamen.addChild(taniHyouji);
```

taniHyouji.taniをtaniHyouji.textに代入して画面に表示されるようにします。文字の装飾のための各値を代入したらSceneに紐付けしましょう。

C. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。画面の中央に「0」が表示されたら成功です。



第13章 ラベルを更新してスコアを表示しよう

[TaniHyouji]単位数のインクリメントと再表示

モルがプリントを獲得したらスコアが増えるようにしましょう。Labelは代入元の変数

(taniHyouji.tani) が変動しても、textの値を更新しなければ表示は変わりません。従ってスコアを増やすにはtaniHyouji.taniの数を増やし、その値をtaniHyouji.textに代入し直す必要があります。

```
054
055     tani.addEventListener("enterframe", function(){
056         if(this.intersect(mol)){
057             gameGamen.removeChild(this);
058             taniHyouji.tani++;
059             taniHyouji.text = taniHyouji.tani.toString();
060
061         }
062         this.y += 4;
063         if(this.y > GAMEN_TATE){
064             gameGamen.removeChild(this);
065         }
066     }
067 }
```

A. スコアを増やす

```
taniHyouji.tani++;
```

taniHyouji.taniを+1します。

B. 表示を更新する

```
taniHyouji.text = taniHyouji.tani.toString();
```

taniHyouji.textにtaniを再代入します。javascriptでは原則型変換の必要はありませんが、Labelに代入するときは明示的に文字列に変換する必要があります。

C. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。画面のスコアが更新されたら成功です。



第14章 キャラクターの動きを制限してゲーム性をつけよう

[Mol]ontouchmoveのx軸制限

モルが画面を縦横無尽に動けるとゲーム性がないので縦方向には動けないようにしましょう。
touchmoveイベントで行っていたy座標の更新のコードをコメントアウトしてください。

```
039     this.frame = 1;  
040 }  
041  
042     this.x = e.x - this.width / 2;  
043     this.y = e.y - this.height / 2;  
044 // this.y = e.y - this.height / 2;  
045 );  
046 gameGamen.addChild(mol);  
047
```

第15章 新しいゲームシーンを作ろう

[TitleGamen]背景の表示

第03章でgameGamenを作成したのと同じ要領でtitleGamenを作成しましょう。背景画像の表示は第07章を参考にすれば可能ですね。

```
077 taniHyouji.moveTo(325, 250);  
078  
079 gameGamen.addChild(taniHyouji);  
080  
var titleGamen = new Scene();  
  
var titleHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);  
titleHaikei.image = core.assets["title.png"];  
titleGamen.addChild(titleHaikei);  
  
081 };  
082  
083 core.start();  
084 };
```

第16章 新しいゲームシーンを表示してみよう

[core.onload]起動後の画面をゲーム画面からタイトル画面へ

第03章ではgameGamenを表示させるためにgameGamenをpushSceneしました。これをtitleGamenに切り替えましょう。

```
021
022 core.onload = function(){
023
024     var gameGamen = new Scene();
025     core.pushScene(gameGamen);
026
027     var gameHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
028     gameHaikei.image = core.assets["game.png"];
029     gameGamen.addChild(gameHaikei);
```

```
078
079     gameGamen.addChild(taniHyouji);
080
081     var titleGamen = new Scene();
082     core.pushScene(titleGamen);
083
084     var titleHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
085     titleHaikei.image = core.assets["title.png"];
086     titleGamen.addChild(titleHaikei);
```

A. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。ゲーム画面ではなくタイトル画面が表示されたら成功です。



第17章 シーン移動をしよう

[TitleGamen]GameGamenへの移動

第16章のままではゲームシーンへ移動できません。画面をタッチしたらゲーム画面へ移動できるようにしましょう。enchant.jsでは画面に触れたときにtouchstartというイベントを発行します。それを受け取ったときに実行する関数にSceneの切り替えを行うコードを記述しましょう。

```
083 var titleHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
084 titleHaikei.image = core.assets["title.png"];
085 titleGamen.addChild(titleHaikei);
086
087     titleGamen.addEventListener("touchstart", function(){
088         core.replaceScene(gameGamen);
089     });
090
091     };
092
093     core.start();
094 }
```

A. touchstartイベントを受け取る

```
titleGamen.addEventListener("touchstart", function(){  
});
```

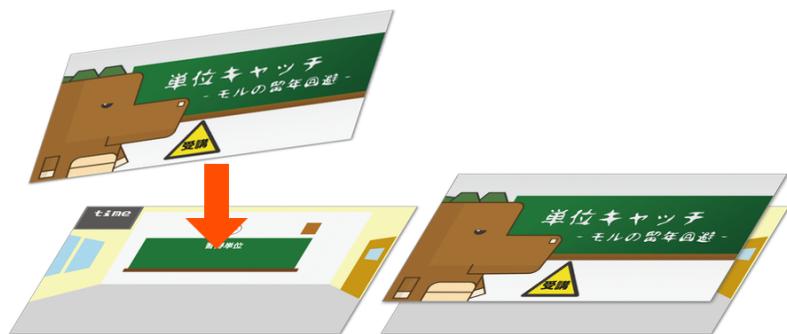
titleGameに触れた時に動作する関数を定義しましょう。

B. Sceneを切り替える

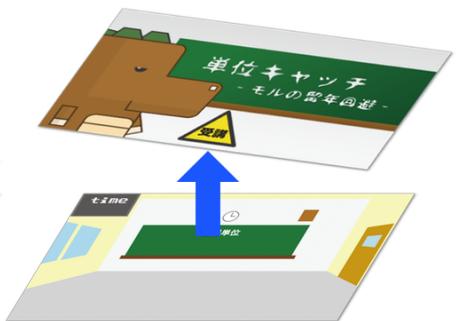
```
core.replaceScene(gameGamen);
```

Sceneの切り替えにはreplaceScene()を用います。pushScene()とは異なるので注意してください。本来pushScene()はSceneを上乗せするときに用います（次項図参照）。上乗せされたSceneはフレームの更新が停止します。一番上のSceneを取り除くにはpopScene()を用います。対してreplaceScene()は表示しているSceneを取り除き指定したSceneに切り替える動作をします（次項図参照）。

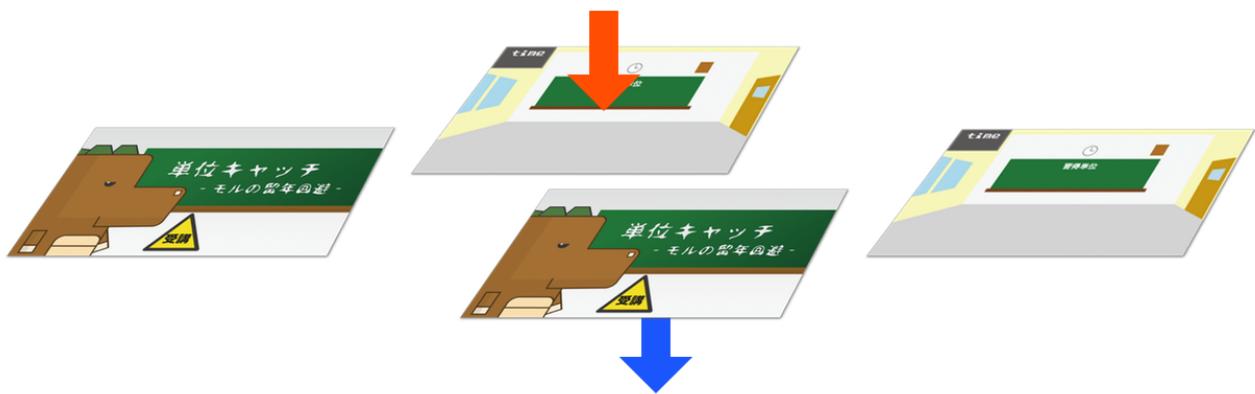
pushScene()



popScene()



replaceScene()



第18章 ラベルを使ってタイマーを作ろう

[GameGamen] タイマの表示

スコア表示と同じようにして残り時間を作ってみましょう。taniHyoujiと違って今回は変数nokoriJikanを外に出しましたが (jikanHyouji.jikan=30ともできる) 、特に意味はありません。

```
087 titleGamen.addEventListener("touchstart", function(){
088     core.replaceScene(gameGamen);
089 });
090
091 var nokoriJikan = 30;
092
093 var jikanHyouji = new Label();
094 jikanHyouji.text = nokoriJikan;
095 jikanHyouji.width = 180;
096 jikanHyouji.textAlignment = "center";
097 jikanHyouji.moveTo(25, 65);
098 jikanHyouji.font = "48px Serif";
099 jikanHyouji.color = "white";
100
101 gameGamen.addChild(jikanHyouji);
102
103 };
104
105 core.start();
106 };
```

A. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。



第19章 ラベルを毎フレーム更新して残り時間を表示しよう

[NokiriJikan] 残り時間のデクリメントと再表示

残り時間の表示ができました。今回は残り時間を減らしていきます。jikanHyoujiに対して、enterframeを受け取った時の関数を実装しましょう。

```
099     jikanHyouji.color = "white";
100
101     gameGamen.addChild(jikanHyouji);
102
102     jikanHyouji.addEventListener("enterframe", function(){
103         nokoriJikan -= 1 / core.fps;
104         this.text = nokoriJikan.toFixed(2).toString();
105     });
106 }
```

A. jikanHyoujiのenterframeイベントを受け取る

```
jikanHyouji.addEventListener("enterframe", function(){
});
```

jikanHyoujiに対してenterframeを受け取った時の関数を実装しました。

B. nokoriJikanを毎フレーム減らそう

```
nokoriJikan -= 1 / core.fps;
this.text = nokoriJikan.toFixed(2).toString();
```

core.fpsでゲームのfpsが取得できます。fpsは一秒間あたりのフレーム数のことなので毎フレームごとにフレーム数分の一引けば、一秒後にちょうどnokoriJikanから1引くことができます。nokoriJikanが変更されたらjikanHyouji.textを更新する必要があります。toFixed()は引数で与えられた数桁数で小数点以下を丸める関数です。今回は小数点以下二桁まで表示したいので引数に2を指定しました。

C. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。残り時間が減っていくようにならたら成功です。



第20章 ゲームオーバーシーンを作ろう

[GameOverGamen]仮背景の表示

第03章でgameGamenを作成したのと同じ要領でgameOverGamenを作成しましょう。背景画像の表示は第07章を参考にすれば可能ですね。

```
104     nokoriJikan -= 1 / core.fps;
105     this.text = nokoriJikan.toFixed(2).toString();
106   });
107
108   var gameOverGamen = new Scene();
109
110   var gameOverHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
111   gameOverHaikei.image = core.assets["game_over1.png"];
112
113   gameOverGamen.addChild(gameOverHaikei);
114
115   core.start();
116 }
```

第21章 条件でシーン移動できるようにしよう

[GameGamen]時間切れでGameOverGamenへの移動

残り時間が0になったらゲームオーバー画面に移動するようにしましょう。gameGamenのenterframeイベントを受け取った時に実行する関数内に以下の条件式を書きます。

```
063       gameGamen.removeChild(this);
064     }
065   });
066 }
067 if(nokoriJikan < 0){
068   core.replaceScene(gameOverGamen);
069 }
070 var taniHyouji = new Label();
071 taniHyouji.tani = 0;
```

第22章 条件で背景画像を変えよう。

[GameOverGamen]スコアに応じて背景画像の切り替え

ゲームオーバー画面での表示内容をスコアの優劣によって変えてみましょう。この章で考えてほしいことはコードがいつ実行されているかです。ゲームオーバー画面での表示をゲーム結果によって変化させたいときコードはいつ実行されるべきでしょう。ゲームのスコアが確定した直後かつゲームオーバー画面が表示される直前が最適です。今回のゲームでは第21章で書いた残り時間判定のコードの部分ですね。ゲームの背景を表示するコードをそこへ移し、スコアの優劣によって背景画像を変更するコードを実装します。

```
064      }
065  });
066  }
067 if(nokoriJikan < 0){
    var gameOverHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);

    if(taniHyouji.tani < 20){
        gameOverHaikei.image = core.assets["game_over1.png"];
    }else{
        gameOverHaikei.image = core.assets["game_over2.png"];
    }

    gameOverGamen.addChild(gameOverHaikei);
068  core.replaceScene(gameOverGamen);
069  }
070  });
071
```

```
108  this.text = nokoriJikan.toFixed(2).toString();
109  });
110
111 var gameOverGamen = new Scene();
112
113 var gameOverHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
114 gameOverHaikei.image = core.assets["game_over1.png"];
115
116 gameOverGamen.addChild(gameOverHaikei);
117
118 };
119
120 core.start();
```

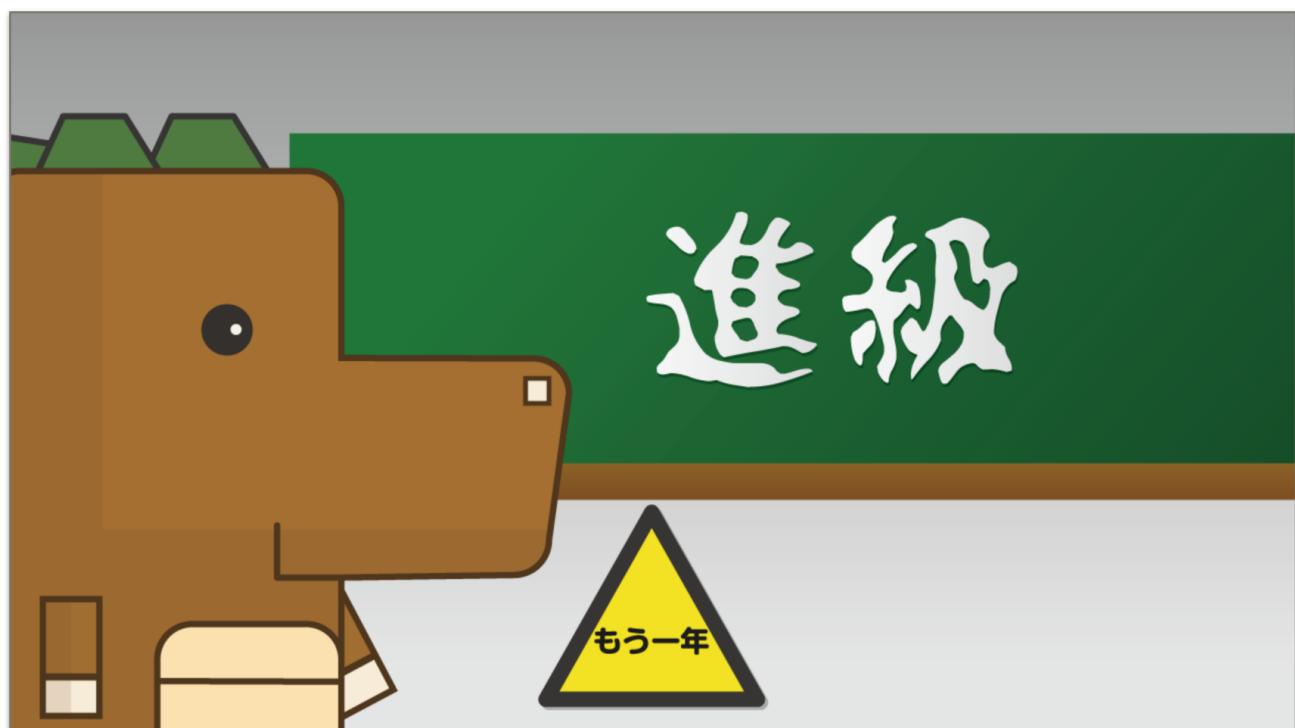
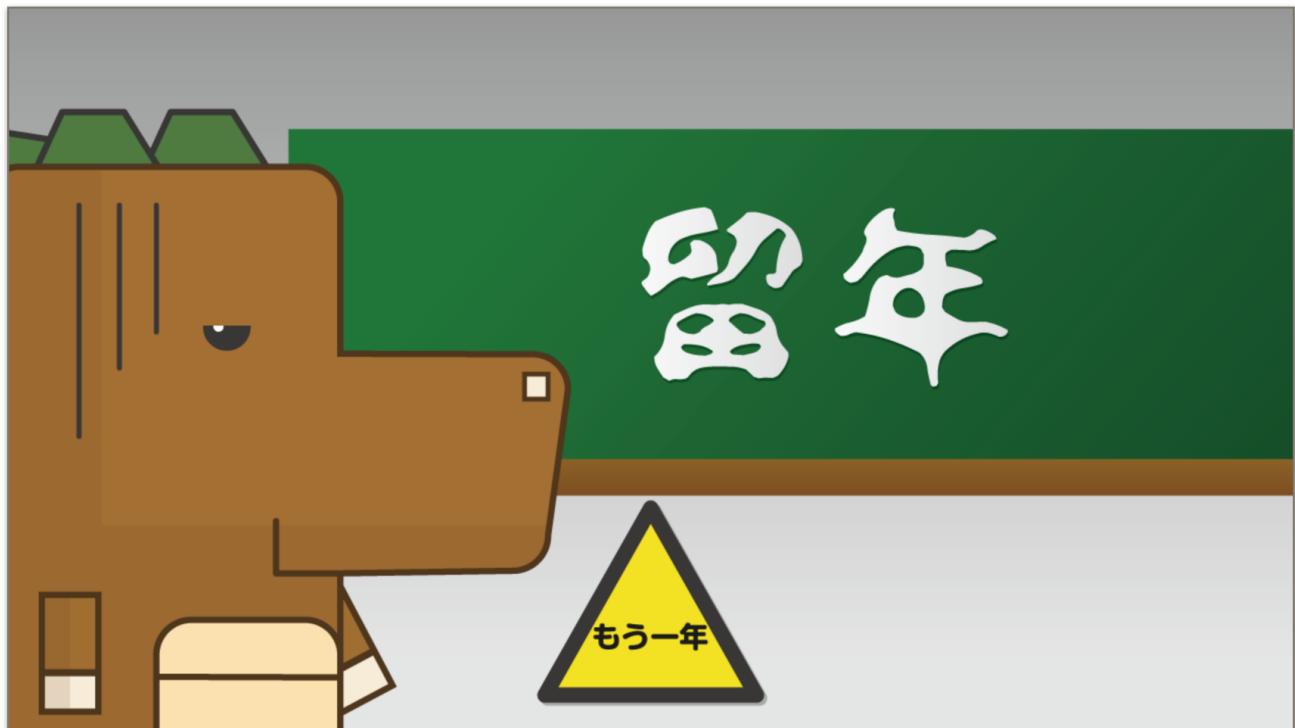
A. スコアで背景画像を変更する

```
if(taniHyouji.tani < 20){
}else{
}
```

スコアが20より低いかそれ以上かで分けました。

B. ゲームを表示してみよう

ブラウザの更新ボタンを押してゲーム画面を再読み込みしましょう。スコアが20以上かそうでないかでゲームオーバー画面が変われば成功です。



付録

コード全文

```
GAMEN_YOKO = 1280;
GAMEN_TATE = 720;

enchant();

var core = {};

window.onload = function(){

    core = new Core(GAMEN_YOKO, GAMEN_TATE);

    var gazou = [];
    gazou.push("title.png");
    gazou.push("game.png");
    gazou.push("game_over1.png");
    gazou.push("game_over2.png");
    gazou.push("mol.png");
    gazou.push("print.png");

    core.preload(gazou);

    core.onload = function(){

        var gameGamen = new Scene();

        var gameHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
        gameHaikei.image = core.assets["game.png"];
        gameGamen.addChild(gameHaikei);

        var mol = new Sprite(151, 169);
        mol.image = core.assets["mol.png"];
        mol.moveTo(300, 500);

        mol.addEventListener("touchmove", function(e){
            if(e.x - this.width / 2 > this.x){
                this.frame = 0;
            }else{
                this.frame = 1;
            }

            this.x = e.x - this.width / 2;
            // this.y = e.y - this.height / 2;
        });

        gameGamen.addChild(mol);

        gameGamen.addEventListener("enterframe", function(){
            if(this.age % 30 == 0){
                var tani = new Sprite(95, 100);
                tani.image = core.assets["print.png"];
            }
        });
    };
}
```

```

tani.moveTo(Math.random() * (GAMEN_YOKO - tani.width), -tani.height);

gameGamen.addChild(tani);

tani.addEventListener("enterframe", function(){
    if(this.intersect(mol)){
        gameGamen.removeChild(this);
        taniHyouji.tani++;
        taniHyouji.text = taniHyouji.tani.toString();
    }
    this.y += 4;
    if(this.y > GAMEN_TATE){
        gameGamen.removeChild(this);
    }
});
});

if(nokoriJikan < 0){
    var gameOverHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);

    if(taniHyouji.tani < 20){
        gameOverHaikei.image = core.assets["game_over1.png"];
    }else{
        gameOverHaikei.image = core.assets["game_over2.png"];
    }

    gameOverGamen.addChild(gameOverHaikei);
    core.replaceScene(gameOverGamen);
}
});

var taniHyouji = new Label();
taniHyouji.tani = 0;
taniHyouji.text = taniHyouji.tani;
taniHyouji.width = 600;
taniHyouji.textAlign = "center";
taniHyouji.font = "80px Serif";
taniHyouji.color = "white";
taniHyouji.moveTo(325, 250);

gameGamen.addChild(taniHyouji);

var titleGamen = new Scene();
core.pushScene(titleGamen);

var titleHaikei = new Sprite(GAMEN_YOKO, GAMEN_TATE);
titleHaikei.image = core.assets["title.png"];
titleGamen.addChild(titleHaikei);

titleGamen.addEventListener("touchstart", function(){
    core.replaceScene(gameGamen);
});

var nokoriJikan = 30;

var jikanHyouji = new Label();
jikanHyouji.text = nokoriJikan;

```

```
jikanHyouji.width = 180;
jikanHyouji.textAlign = "center";
jikanHyouji.moveTo(25, 65);
jikanHyouji.font = "48px Serif";
jikanHyouji.color = "white";

gameGamen.addChild(jikanHyouji);

jikanHyouji.addEventListener("enterframe", function(){
    nokoriJikan -= 1 / core.fps;
    this.text = nokoriJikan.toFixed(2).toString();
});

var gameOverGamen = new Scene();

};

core.start();
};
```

作成者：高橋 一貴
作成日：2013年12月9日月曜日