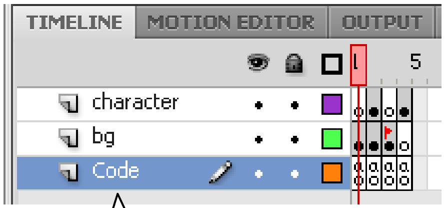
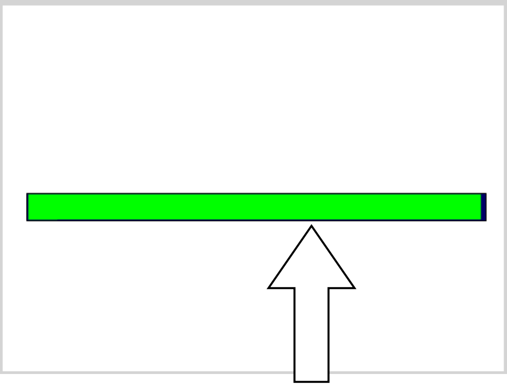
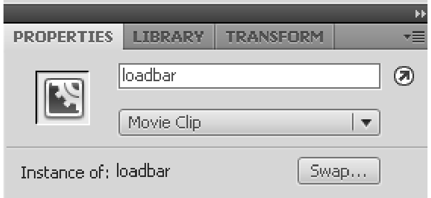
Move4



```
stop();

onEnterFrame = function () {

		loadbar._xscale = _root.getBytesLoaded()/_root.getBytesTotal()*100;

};

if ((_root.getBytesLoaded()/_root.getBytesTotal()*100)>=100)  {

		_root.nextFrame();

		loadbar.play();

}
```





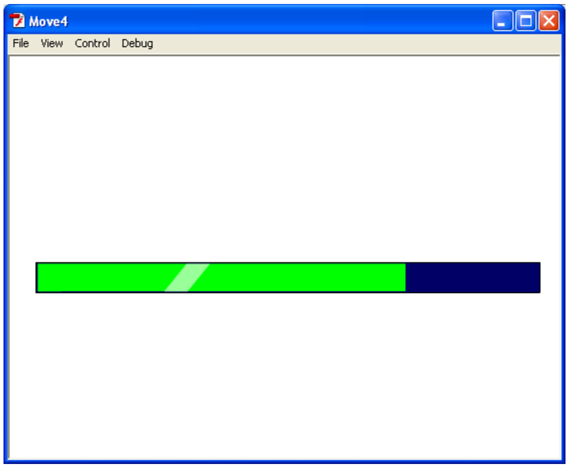
loadbar

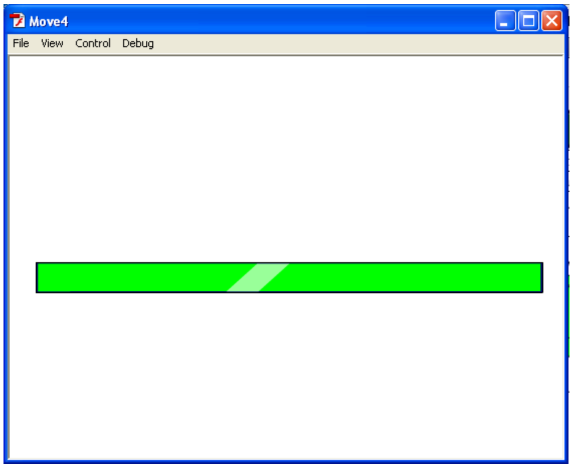
Instance of: loadbar

This is the loading screen in action.

Behind the green loadbar is a blue bar that shows how far the green bar will go.

The orientation of the loadbar is to the middle left.

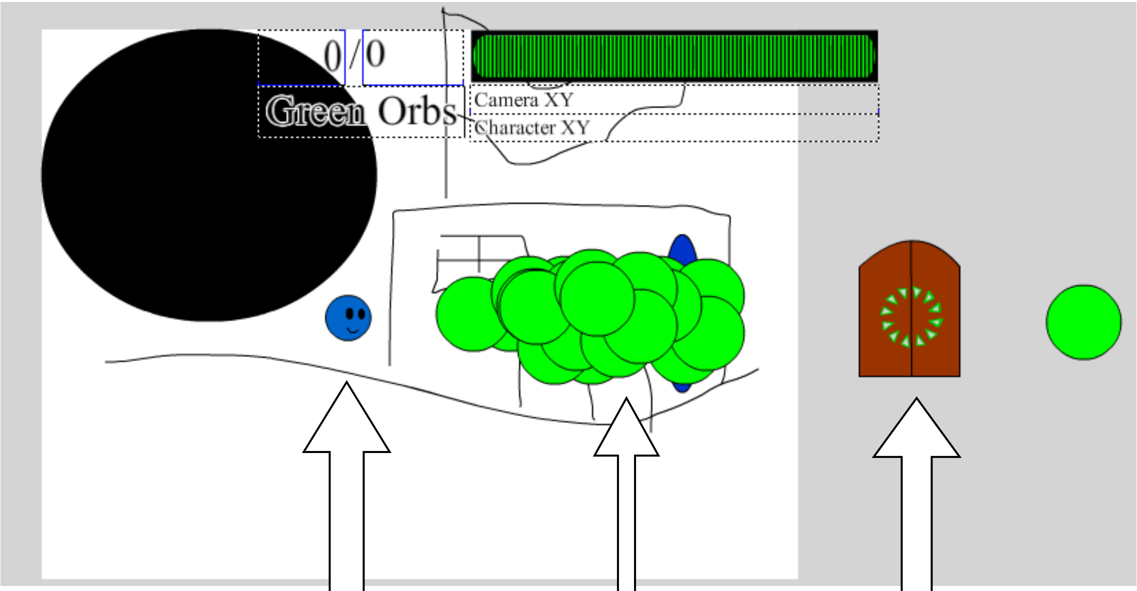


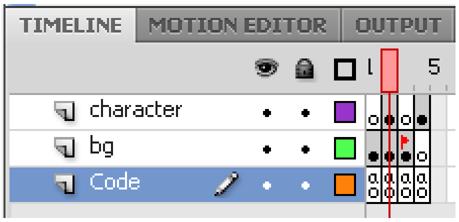


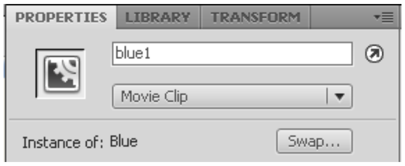
When it's loaded, it goes to the next frame automatically.

This is the first level. In this frame, several functions and variables are made to make the blue smiley face move around, collect the green orbs, and exit through the door.

You can use the arrow keys, WASD keys, shift, and the ctrl button.



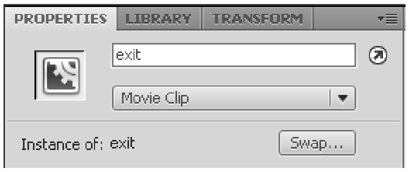




blue1

Movie Clip

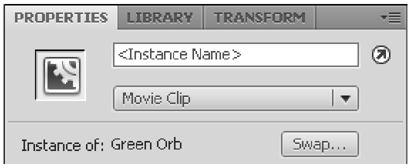
Instance of: Blue



exit

Movie Clip

Instance of: exit



<Instance Name>

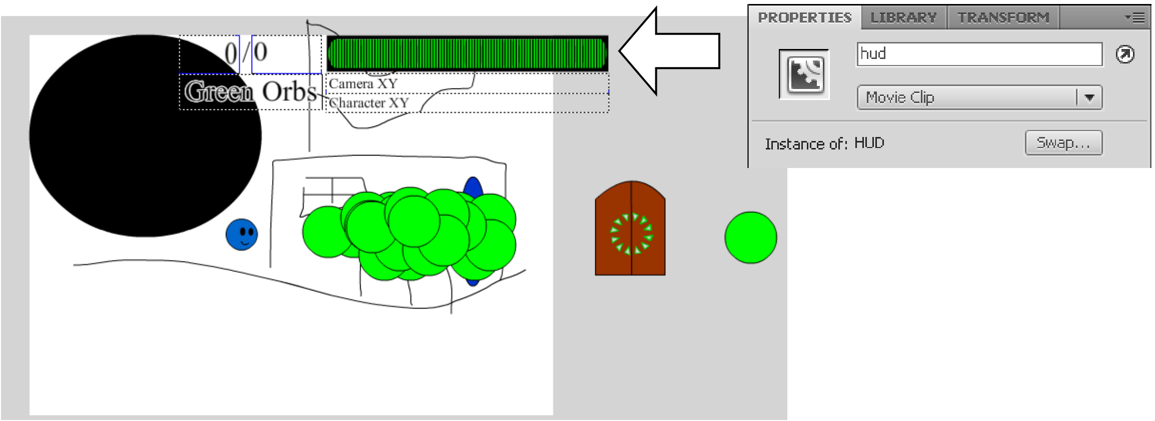
Movie Clip

Instance of: Green Orb

The Green Orbs aren't manually given an instance name. Instead, they are dynamically assigned names with the code inside them.

```
/*

Adam Schachte

February 28,2012

ActionScript 2.0


You can move the camera freely when you hold the shift button.

When you move the character with the arrow keys, the camera follows him.

Left and right make him walk. Up and down make him rotate.

Ctrl makes him attack. The movement isn't choppy anymore.

The drawing of a building just makes it easy to see the guy move.

I added a loading bar, a HUD, green orbs, particles, and organized the library.

Also, the WASD keys work, but when you use them, the walking animation doesn't work.

*/

stop();


blue1.walkspeed=5;
```

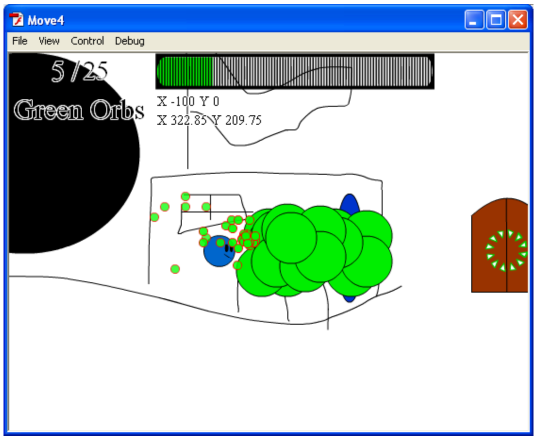
```
blue1.attack=false;

blue1.jump=false;

blue1.crouch=false;


upButton=false;

downButton=false;

leftButton=false;

rightButton=false;

attackButton=false;

debugButton=false;


var greenOrbs=0;

var greenOrbMax=0;

var particles=0;
```



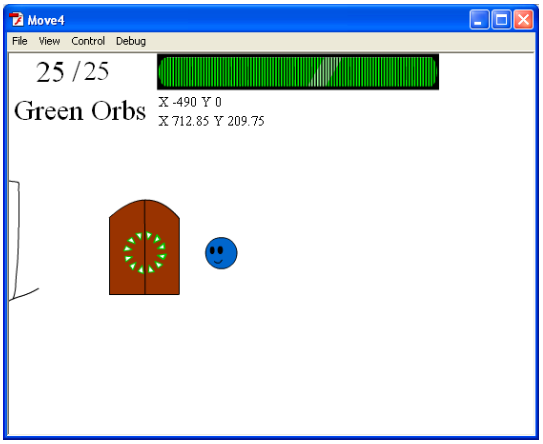
hud

Movie Clip

Instance of: HUD



```
5 /25
Green Orbs
X -100 Y 0
X 322.85 Y 209.75
```



```
25 /25
Green Orbs
X -490 Y 0
X 712.85 Y 209.75
```

```
debugMove = function(){

          if (Key.isDown(Key.UP)&&Key.isDown(Key.DOWN)&&Key.isDown(Key.LEFT)&&Key.isDown(Key.RIGHT)){

                              trace("Up, Down, Left, and Right");//all 4 directions.

                    }

          else if (Key.isDown(Key.UP)&&Key.isDown(Key.DOWN)&&Key.isDown(Key.LEFT)){

                              trace("Up, Down, and Left");//3 directions. Missing right.

                    }

                    else if (Key.isDown(Key.UP)&&Key.isDown(Key.DOWN)&&Key.isDown(Key.RIGHT)){

                              trace("Up, Down, and Right");//3 directions.  Missing left.

                    }

                    else if (Key.isDown(Key.UP)&&Key.isDown(Key.LEFT)&&Key.isDown(Key.RIGHT)){

                              trace("Up, Left, and Right");//3 directions. Missing down.

                              _y=_y+10;//moves stage up

                    }

                    else if (Key.isDown(Key.DOWN)&&Key.isDown(Key.LEFT)&&Key.isDown(Key.RIGHT)){
```

```
    trace("Down, Left, and Right");//3 directions.  Missing up.

    _y=_y-10;

}

else {//Beginning of the 2 directional code.

    if (Key.isDown(Key.UP)&&Key.isDown(Key.DOWN)){

    trace("Up and Down");//2 directions.  1st up.  1st down.

}

    else if (Key.isDown(Key.UP)&&Key.isDown(Key.LEFT)){

        trace("Up and Left");//2 directions.  2nd up.  1st left.

        _y=_y+10;//moves stage up

        _x=_x+10;//moves stage left

    }

    else if (Key.isDown(Key.UP)&&Key.isDown(Key.RIGHT)){

        trace("Up and Right");//2 directions.  3rd up.  1st right.

        _y=_y+10;//moves stage up

        _x=_x-10;//moves stage right

    }

    else if (Key.isDown(Key.DOWN)&&Key.isDown(Key.LEFT)){

        trace("Down and Left");//2 directions.  2nd down.  2nd left.

        _y=_y-10;//moves stage down

        _x=_x+10;//moves stage left

    }

    else if (Key.isDown(Key.DOWN)&&Key.isDown(Key.RIGHT)){

        trace("Down and Right");//2 directions.  3rd down.  2nd right.

        _y=_y-10;//moves stage down

        _x=_x-10;//moves stage right

    }

    else if (Key.isDown(Key.LEFT)&&Key.isDown(Key.RIGHT)){

        trace("Left and Right");//2 directions.  3rd left.  3rd right.

    }

    else {//Beginning of the 1 direction.
```

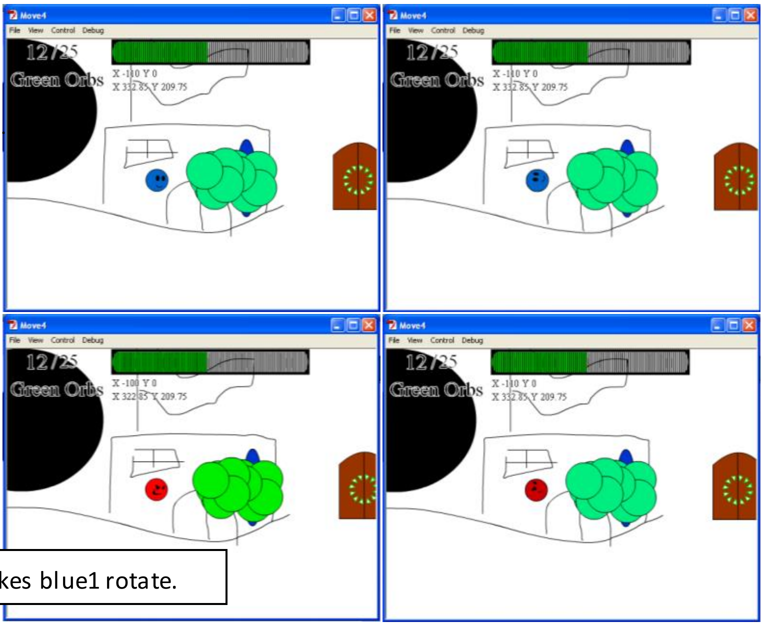
```
                if (Key.isDown(Key.UP)){
                    trace("Up");
                    //blue1.y=blue1.y+10;
                    _y=_y+10;//moves stage up
                }
                else if (Key.isDown(Key.DOWN)){
                    trace("Down");
                    _y=_y-10;//moves stage down
                }
                else if (Key.isDown(Key.LEFT)){
                    trace("Left");
                    _x=_x+10;//moves stage left
                }
                else if (Key.isDown(Key.RIGHT)){
                    trace("Right");
                    _x=_x-10;//moves stage right
                }
            }//end of the 1 directional code.
        }//end of the else statement that started the 2 directional code.
}//end of the debugMove function.
```

The Listen function makes it so the Arrow Keys and WASD Keys work. At first, it was intended to let the user choose between them by clicking on a button.

Key.isDown(65) is using the SCII code for “a”.

```
Listen=function(){
    if (Key.isDown(Key.LEFT) or Key.isDown(65)){
        leftButton=true;
    }//end of if left is pressed statement.
    else {
        leftButton=false;
    }
    if (Key.isDown(Key.RIGHT) or Key.isDown(68)){
        rightButton=true;
    }//end of if right is pressed statement.
    else {
        rightButton=false;
    }
    if (Key.isDown(Key.UP) or Key.isDown(87)){
        upButton=true;
    }//end of if up is pressed statement.
    else {
        upButton=false;
    }
    if (Key.isDown(Key.DOWN) or Key.isDown(83)){
        downButton=true;
    }//end of if down is pressed statement.
    else {
        downButton=false;
    }
    if (Key.isDown(Key.CONTROL)){
        attackButton=true;
    }//end of if alt is pressed statement.
    else {
        attackButton=false;
    }
    if (Key.isDown(Key.SHIFT)){
        debugButton=true;
    }//end of if shift is pressed statement.
    else {
        debugButton=false;
    }
}//end of Listen=function()



Move=function(){
    if (debugButton){//beginning of debug movement with shift button.
        debugMove();
//      if(!attack){
//          blue1.gotoAndStop("stance");
//      }
    }
    else {
        if (leftButton and rightButton){
            trace("character left and right");
            if (!attack) {
                blue1.gotoAndStop("stance");
            }
        }//end of if left and right are pressed statement.
        else {
            if (leftButton and !rightButton){
                blue1._rotation=0;
                blue1._xscale=-100;
                blue1._x-=blue1.walkspeed;
                trace("character left");
                if (!attack) {
                    blue1.gotoAndStop("walk");
                }
            }//end of if left is pressed statement.
            if (!leftButton and rightButton){
                blue1._rotation=0;
                blue1._xscale=100;
                blue1._x+=blue1.walkspeed;
                trace("character right");
                if (!attack) {
                    blue1.gotoAndStop("walk");
                }
```

This makes it so that the smiley doesn’t keep walking while in debug mode. Since it’s been commented out, he will still do it.



```
            }//end of if right is pressed statement.
        }//end of else statement.
//Left/right movement should be indepentent of up/down movement.
        if (downButton && upButton){
            trace("character up and down");
        }//end of if up and down are pressed statement.
        else if (downButton){
            trace ("character down");
            blue1._rotation-=5;
```

This makes blue1 rotate.

```
        }//end of else if down is pressed statement.
        else if (upButton){
            trace ("character up");
            blue1._rotation+=5;
        }//end of else if up is pressed statement.
        else if (attackButton){
            attack=true;
            blue1.gotoAndStop("attack");
            trace("character attack");
        }//end of if else the attack button is pressed statement.
        if (attack){
            blue1.gotoAndStop("attack");
            trace("character attack");
        }
    }//end of character movement.
}//end of Move=function()
this.onEnterFrame = function (){
```

The onEnterFrame function runs the code once every frame. I have this Flash file to 30 Frames Per Second.

```
    Listen();
    Move();
    //camera movement.
        if (!debugButton){
```

```
        _root._x=-(blue1._x)+222.85

        _root._y=-(blue1._y)+209.75
```

I got the last numbers by looking at the properties of blue1 in the first level. Without them, he would be in the top left corner.

```
    }

    hud._x=-_root._x;

    hud._y=-_root._y;
```

This keeps the Heads Up Display in the top left corner of the screen. If it doesn't get negative the root value, it will go in the wrong direction when the character moves.

```
    if (greenOrbs/greenOrbMax>=1){

        _root.hud.loadbar.play();

    }

    else {

        _root.hud.loadbar.gotoAndStop(1);

    }
```

I reused the loadbar to show the progress on the orb collecting.

This makes the loadbar's shine animation play only when the play has collected all the orbs. Since the else statement didn't exist in the loading screen, it will always play there.

```
//green orbs

    if (_root.blue1.hitTest(exit)==true and greenOrbs>=greenOrbMax){

        gotoAndStop("secret");

        trace("exit")

    }
```

"secret" is the name of the next frame.

You can only go through the exit when all the green orbs are collected.

```
    for (var greenOrbNum=1;greenOrbNum<=greenOrbMax;greenOrbNum++){

        if (blue1.hitTest(_root["greenOrb"+greenOrbNum])==true){

            for (var i=1;i<=7;i++){
```

Linkage Name

New Name

New Depth

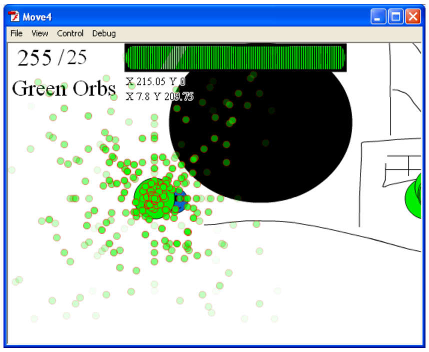
```
                part = _root.attachMovie("particle","particle"+particles,this.getNextHighestDepth());

                part._x=_root["greenOrb"+greenOrbNum]._x;

                part._y=_root["greenOrb"+greenOrbNum]._y;


            }
```

This for loop makes 7 particles, gives them the same variable name "part", and gives them the coordinates of the green orb they came from.

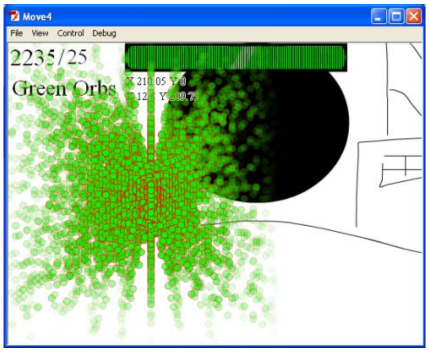
```
            greenOrbs++;

            _root["greenOrb"+greenOrbNum]._x=-20;

        }//end of hit test if statement.

    }//end of for loop.
```

This doesn't get rid of the green orbs. Instead, it moves them over to the left. This makes it possible to get the same orbs over and over.

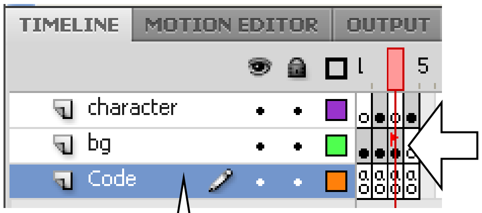
```
                    trace(particles+" particles");

} //end of function
```



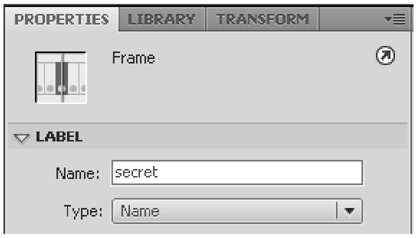
Continuously getting 1 orb.



Continuously getting 21 orbs.



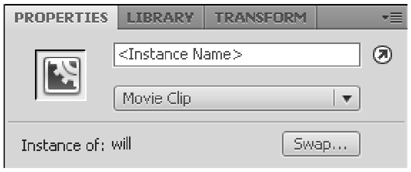
```
stop();

_root._x=_root._y=0;
```



Name: secret

Type: Name

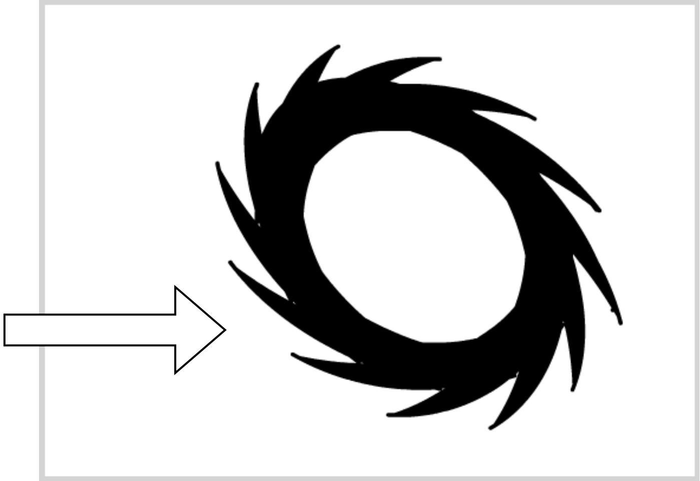
The stage starts where it left off in the last frame. It needs to go back to its original coordinates to see the animation.

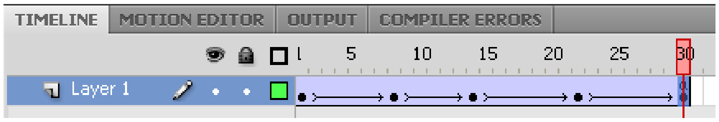


<Instance Name>

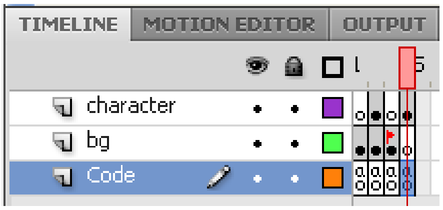
Movie Clip

Instance of: will

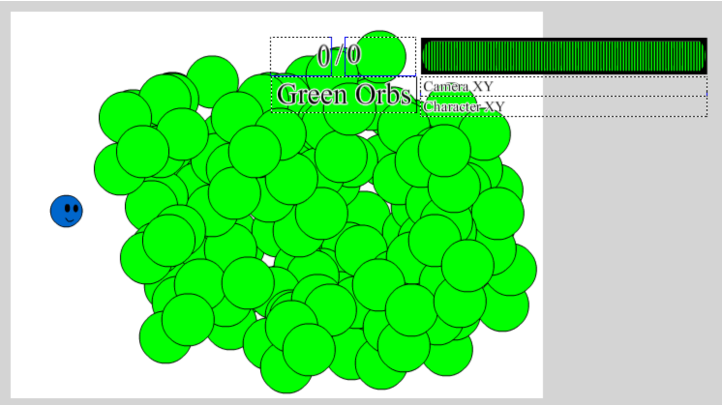




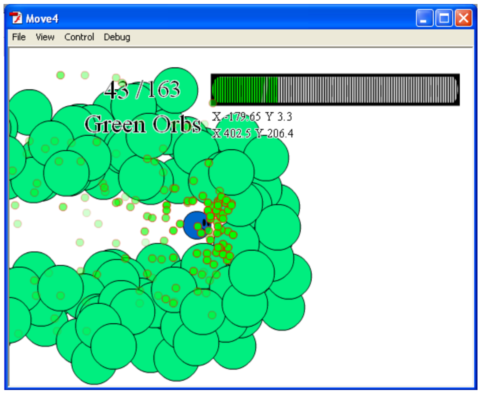
```
_root.nextFrame();
```

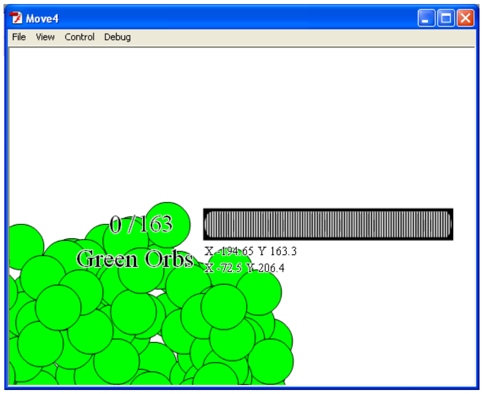


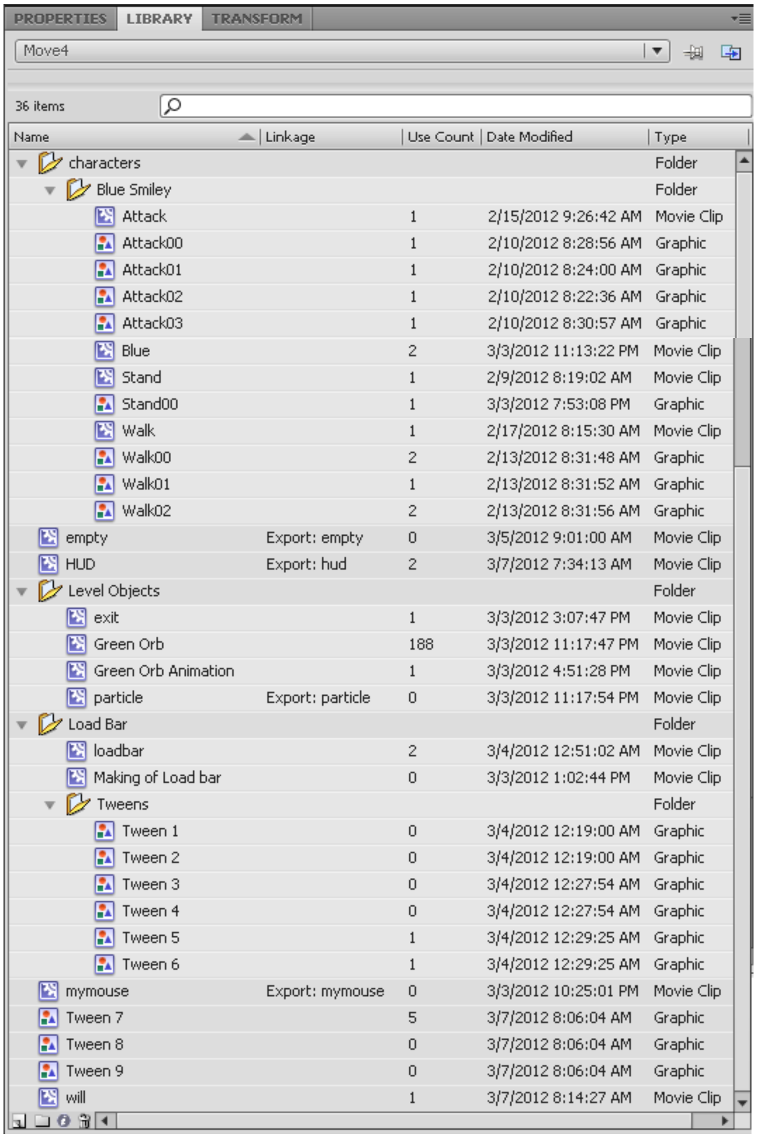
Same code as the first level.



Since the HUD wasn't given an instance name this time, it doesn't follow the camera and it always plays the shine animation.

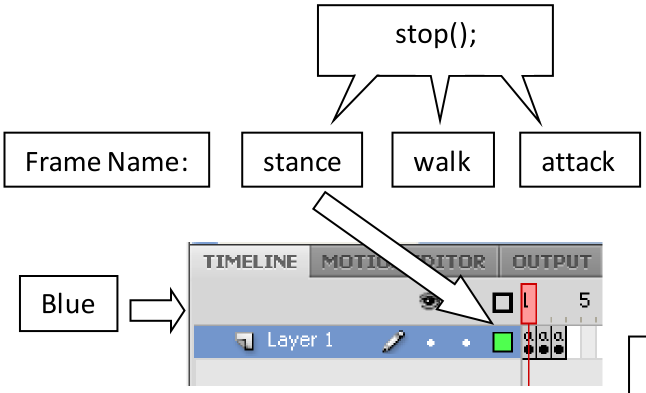






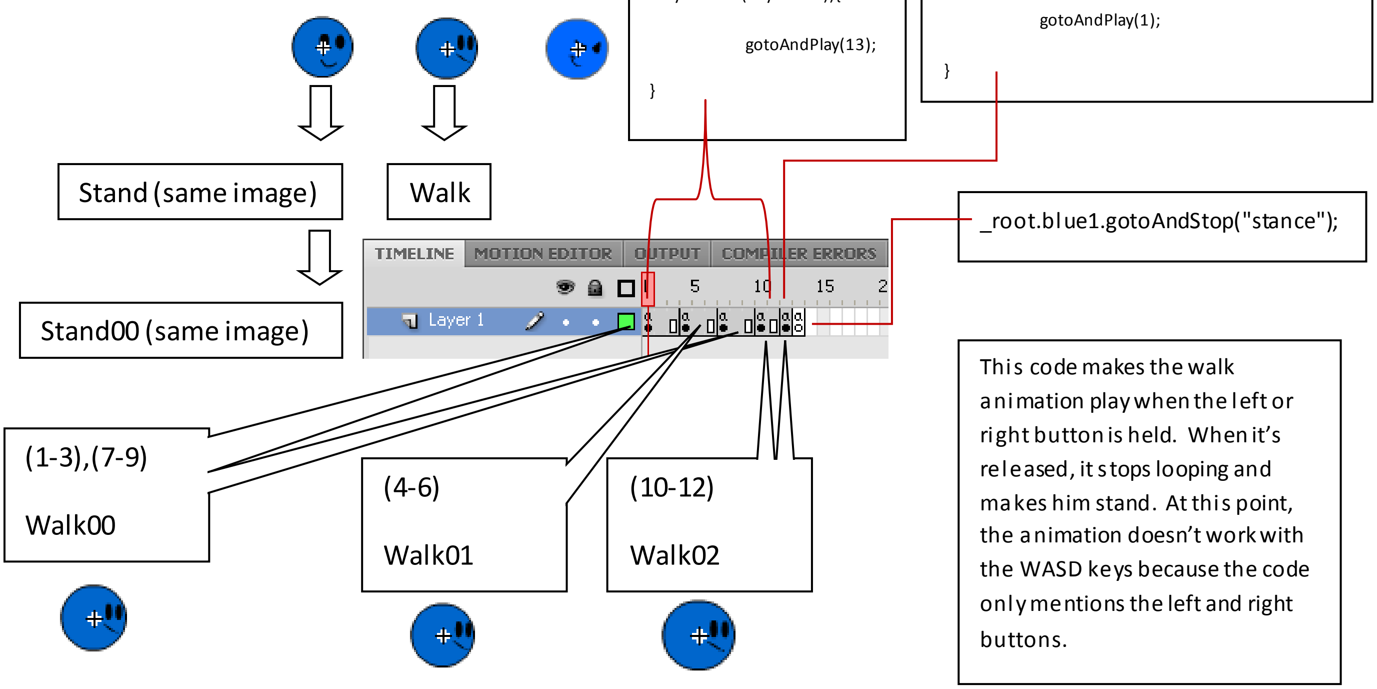
36 items

Name	Linkage	Use Count	Date Modified	Type
characters				Folder
Blue Smiley				Folder
Attack		1	2/15/2012 9:26:42 AM	Movie Clip
Attack00		1	2/10/2012 8:28:56 AM	Graphic
Attack01		1	2/10/2012 8:24:00 AM	Graphic
Attack02		1	2/10/2012 8:22:36 AM	Graphic
Attack03		1	2/10/2012 8:30:57 AM	Graphic
Blue		2	3/3/2012 11:13:22 PM	Movie Clip
Stand		1	2/9/2012 8:19:02 AM	Movie Clip
Stand00		1	3/3/2012 7:53:08 PM	Graphic
Walk		1	2/17/2012 8:15:30 AM	Movie Clip
Walk00		2	2/13/2012 8:31:48 AM	Graphic
Walk01		1	2/13/2012 8:31:52 AM	Graphic
Walk02		2	2/13/2012 8:31:56 AM	Graphic
empty	Export: empty	0	3/5/2012 9:01:00 AM	Movie Clip
HUD	Export: hud	2	3/7/2012 7:34:13 AM	Movie Clip
Level Objects				Folder
exit		1	3/3/2012 3:07:47 PM	Movie Clip
Green Orb		188	3/3/2012 11:17:47 PM	Movie Clip
Green Orb Animation		1	3/3/2012 4:51:28 PM	Movie Clip
particle	Export: particle	0	3/3/2012 11:17:54 PM	Movie Clip
Load Bar				Folder
loadbar		2	3/4/2012 12:51:02 AM	Movie Clip
Making of Load bar		0	3/3/2012 1:02:44 PM	Movie Clip
Tweens				Folder
Tween 1		0	3/4/2012 12:19:00 AM	Graphic
Tween 2		0	3/4/2012 12:19:00 AM	Graphic
Tween 3		0	3/4/2012 12:27:54 AM	Graphic
Tween 4		0	3/4/2012 12:27:54 AM	Graphic
Tween 5		1	3/4/2012 12:29:25 AM	Graphic
Tween 6		1	3/4/2012 12:29:25 AM	Graphic
mymouse	Export: mymouse	0	3/3/2012 10:25:01 PM	Movie Clip
Tween 7		5	3/7/2012 8:06:04 AM	Graphic
Tween 8		0	3/7/2012 8:06:04 AM	Graphic
Tween 9		0	3/7/2012 8:06:04 AM	Graphic
will		1	3/7/2012 8:14:27 AM	Movie Clip



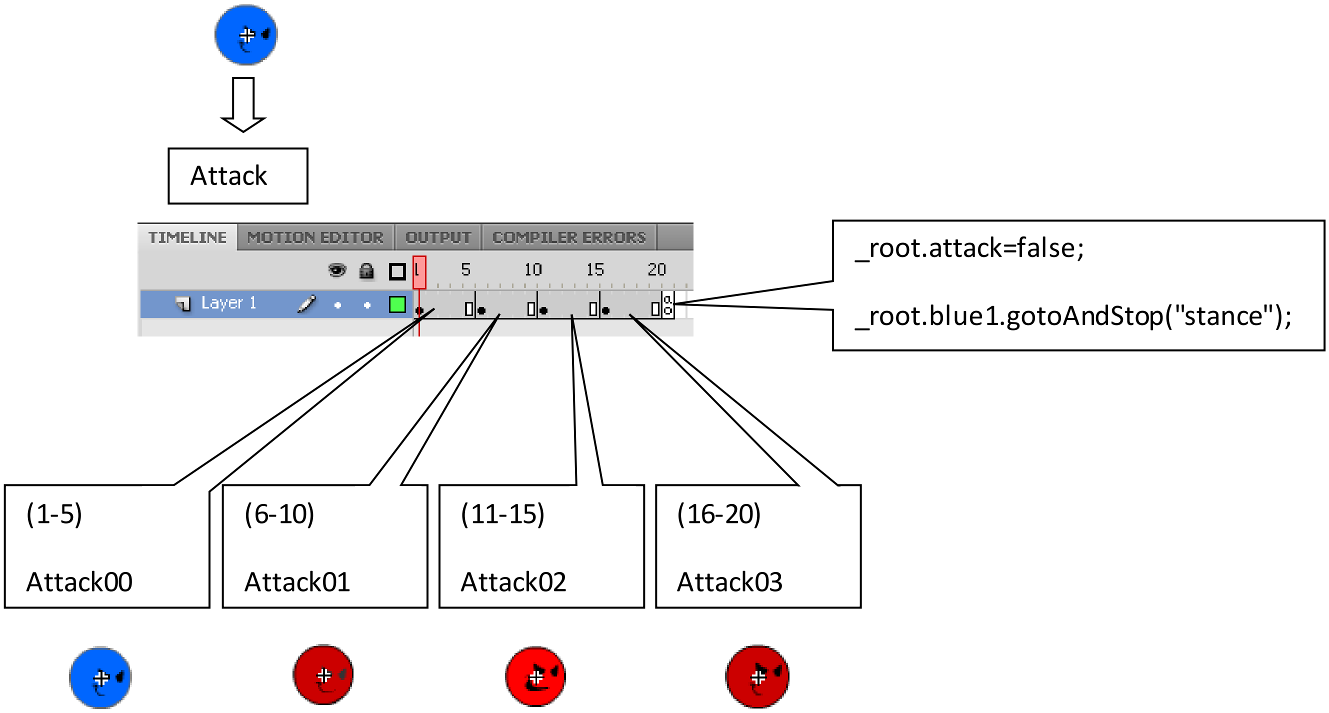
```
if (!Key.isDown(Key.LEFT)and !Key.isDown(Key.RIGHT)){

        gotoAndPlay(13);

}
```

```
if (Key.isDown(Key.LEFT)or  Key.isDown(Key.RIGHT)){

        gotoAndPlay(1);

}
```



```
_root.blue1.gotoAndStop("stance");
```

This code makes the walk animation play when the left or right button is held. When it's released, it stops looping and makes him stand. At this point, the animation doesn't work with the WASD keys because the code only mentions the left and right buttons.



Attack

```
_root.attack=false;

_root.blue1.gotoAndStop("stance");
```

(1-5)

Attack00

(6-10)

Attack01

(11-15)

Attack02

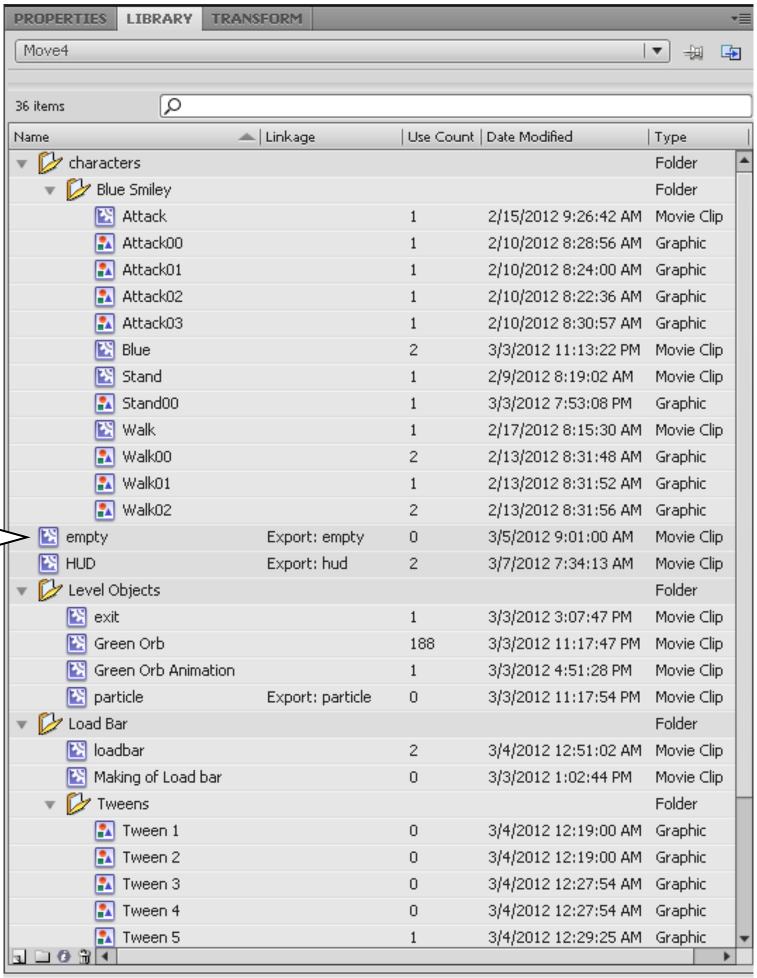
(16-20)

Attack03

The "empty" movie clip was meant to be used for particles. The empty movie clip would spin, giving the particles a cool affect. There's only one frame, but it has some code on it.

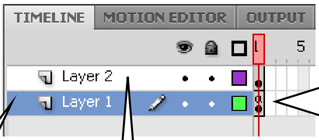
```
if(this._alpha>=0){

        this.removeMovieClip();

}
```

This code only seems to work with movie clips that have been dynamically placed onto the stage. I could never get it to place the particles inside the movie clip, and when I placed them in manually, they never deleted. This would make a function tell me there were negative 240 particles on the screen, and the number would keep going down. The empty movie clip is never used in the code.

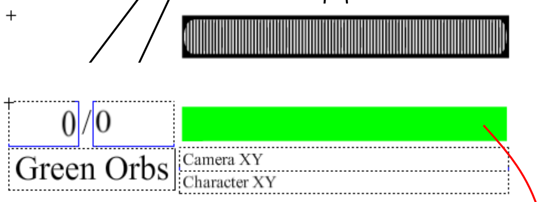


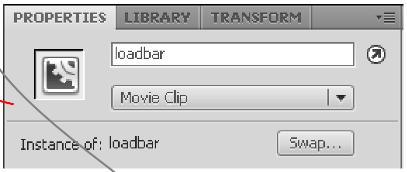
Name	Linkage	Use Count	Date Modified	Type
characters				Folder
Blue Smiley				Folder
Attack		1	2/15/2012 9:26:42 AM	Movie Clip
Attack00		1	2/10/2012 8:28:56 AM	Graphic
Attack01		1	2/10/2012 8:24:00 AM	Graphic
Attack02		1	2/10/2012 8:22:36 AM	Graphic
Attack03		1	2/10/2012 8:30:57 AM	Graphic
Blue		2	3/3/2012 11:13:22 PM	Movie Clip
Stand		1	2/9/2012 8:19:02 AM	Movie Clip
Stand00		1	3/3/2012 7:53:08 PM	Graphic
Walk		1	2/17/2012 8:15:30 AM	Movie Clip
Walk00		2	2/13/2012 8:31:48 AM	Graphic
Walk01		1	2/13/2012 8:31:52 AM	Graphic
Walk02		2	2/13/2012 8:31:56 AM	Graphic
empty	Export: empty	0	3/5/2012 9:01:00 AM	Movie Clip
HUD	Export: hud	2	3/7/2012 7:34:13 AM	Movie Clip
Level Objects				Folder
exit		1	3/3/2012 3:07:47 PM	Movie Clip
Green Orb		188	3/3/2012 11:17:47 PM	Movie Clip
Green Orb Animation		1	3/3/2012 4:51:28 PM	Movie Clip
particle	Export: particle	0	3/3/2012 11:17:54 PM	Movie Clip
Load Bar				Folder
loadbar		2	3/4/2012 12:51:02 AM	Movie Clip
Making of Load bar		0	3/3/2012 1:02:44 PM	Movie Clip
Tweens				Folder
Tween 1		0	3/4/2012 12:19:00 AM	Graphic
Tween 2		0	3/4/2012 12:19:00 AM	Graphic
Tween 3		0	3/4/2012 12:27:54 AM	Graphic
Tween 4		0	3/4/2012 12:27:54 AM	Graphic
Tween 5		1	3/4/2012 12:29:25 AM	Graphic

HUD



```
this.onEnterFrame=function(){
        collectedOrbText=_root.greenOrbs;
        orbMaxText=_root.greenOrbMax;
        cameraxytext="X "+_root._x+" Y "+_root._y;
        characterxytext="X "+_root.blue1._x+" Y "+_root.blue1._y;
        loadbar._xscale=(_root.greenOrbs/_root.greenOrbMax)*58.6746;
        if (loadbar._xscale>58.6746){
                loadbar._xscale=58.6746;
        }//end of if statement.
}//end of function.
```





loadbar

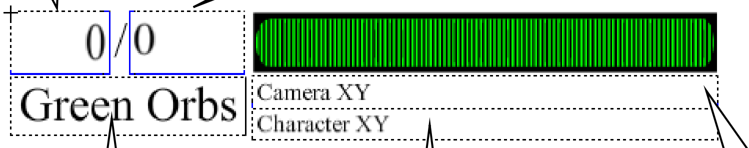
Movie Clip

Instance of: loadbar

Swap...

Static Text

(It has no variable name)



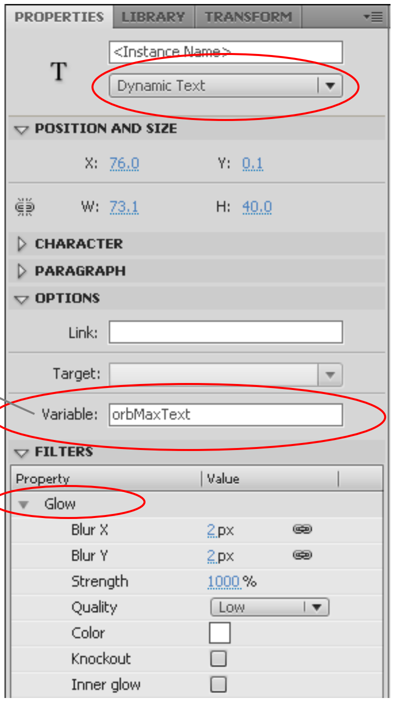
collectedOrbText

orbMaxText

orbText

characterxytext

cameraxytext



<Instance Name>

Dynamic Text

POSITION AND SIZE

X: 76.0 Y: 0.1

W: 73.1 H: 40.0

CHARACTER

PARAGRAPH

OPTIONS

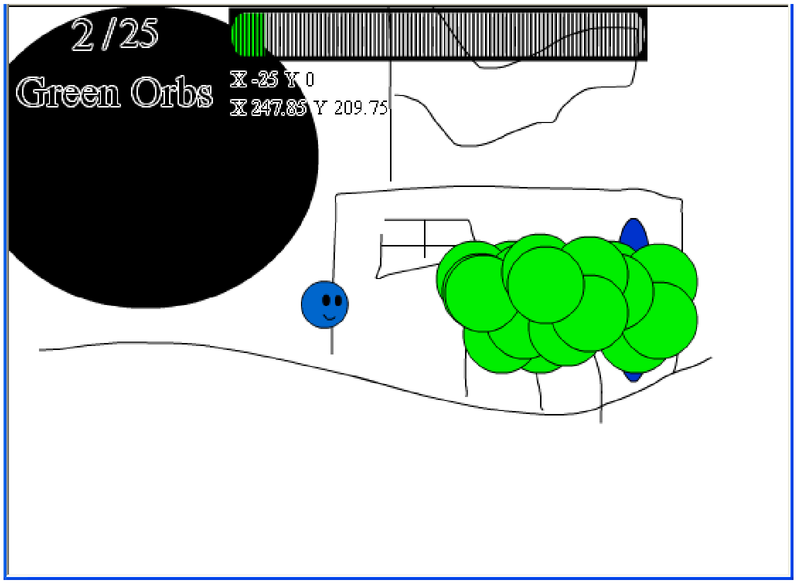
Link:

Target:

Variable: orbMaxText

FILTERS

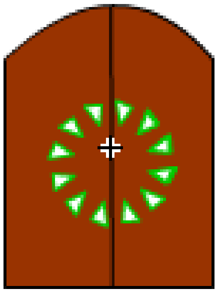
Property	Value
Glow	
Blur X	2 px
Blur Y	2 px
Strength	1000 %
Quality	Low
Color	□
Knockout	□
Inner glow	□



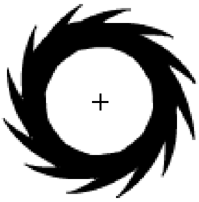
The glow filter is an easy way to make the text visible against a black background. All the text here has it.

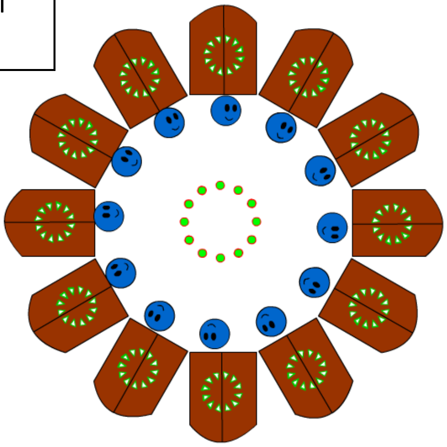
The loadbar has been squished to 58.6746% it's normal length on the x axis. That's why that number is mentioned in the code above.

exit

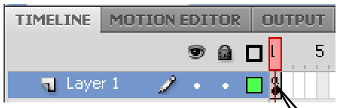


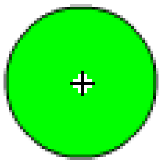
This movie clip only has one frame, one layer, and has no code inside. One green arrow was drawn and was duplicated with the Transform Window. For this, it was set at 30 degrees. Transform can be used to make lots of cool stuff.





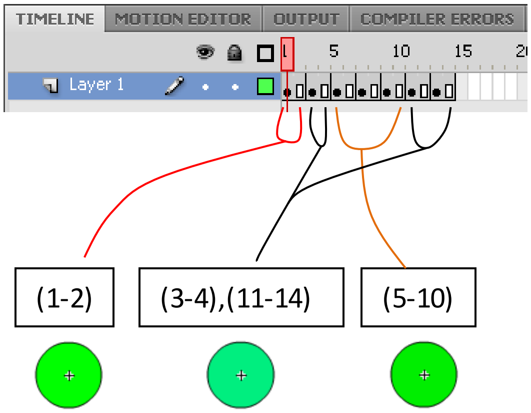
Green Orb





```
_root.greenOrbMax++;

_name = "greenOrb"+_root.greenOrbMax;
```

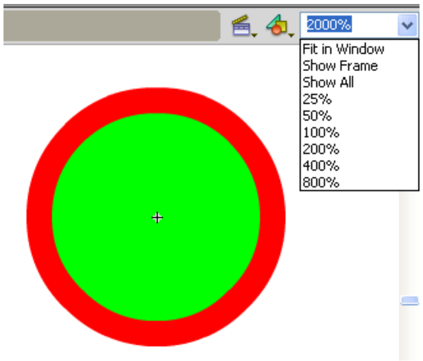
Green Orb Animation



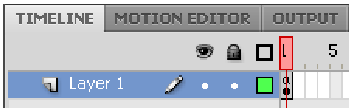
We only want to use this code one time. Otherwise, the number of orbs you need will keep growing faster than you can get them.

That's why it only has one frame. It doesn't loop.

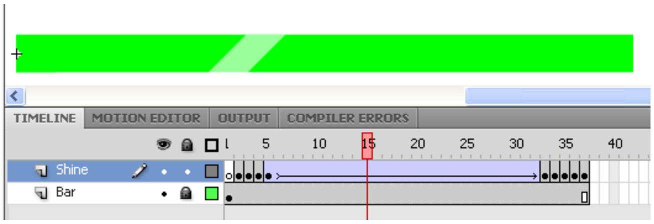
particle



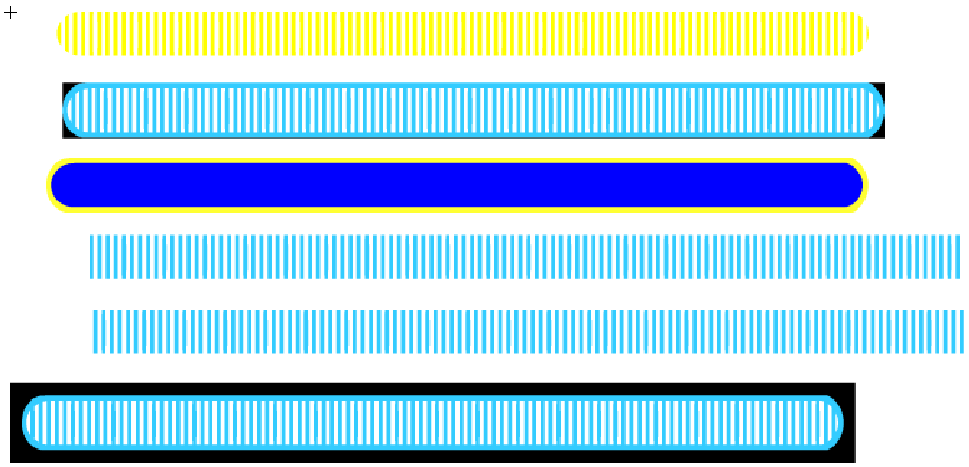
```
xSpeed=random(10)-5;

ySpeed=random(10)-5;

_root.particles++;

onEnterFrame=function(){

        _x+=xSpeed;

        _y+=ySpeed;

        _alpha-=2;

        if(_alpha<=0){

                _root.particles--;

                this.removeMovieClip();

        }//end of if _alpha<=0 statement.

}//end of function
```



loadbar



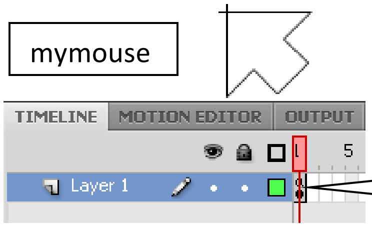
Basically a green bar with a classic tween over it to make it look like it's shining. I manually did the parts where the shine would come off the edges. This ended up making 6 tween graphics. I thought deleting them might mess up the animation, so I stuck them in their own folder.



Making of Load bar

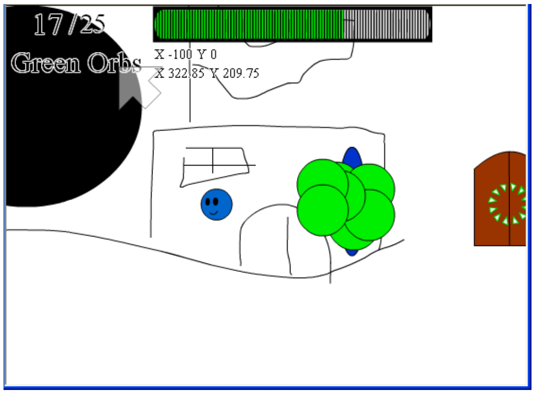
I used these to make the loadbar. I didn't want to throw them away, so I put it in its own movie clip. It's never used.

mymouse



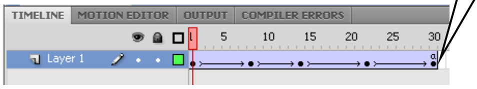
```
onEnterFrame=function(){

        this._x=_root._xmouse;

        this._y=_root._ymouse;

        Mouse.hide();

        this._alpha=50;

}
```

When put on the stage, this will get the xy coordinates of the user's normal mouse, hide the user's normal mouse, then turn this custom mouse see through. When the flash file becomes super laggy, the custom mouse stops moving momentarily and becomes a hassle. When the character walks, it makes the mouse move over a few pixles. If you put the mouse on level 1 and not on level 2, the regular mouse will not reappear.

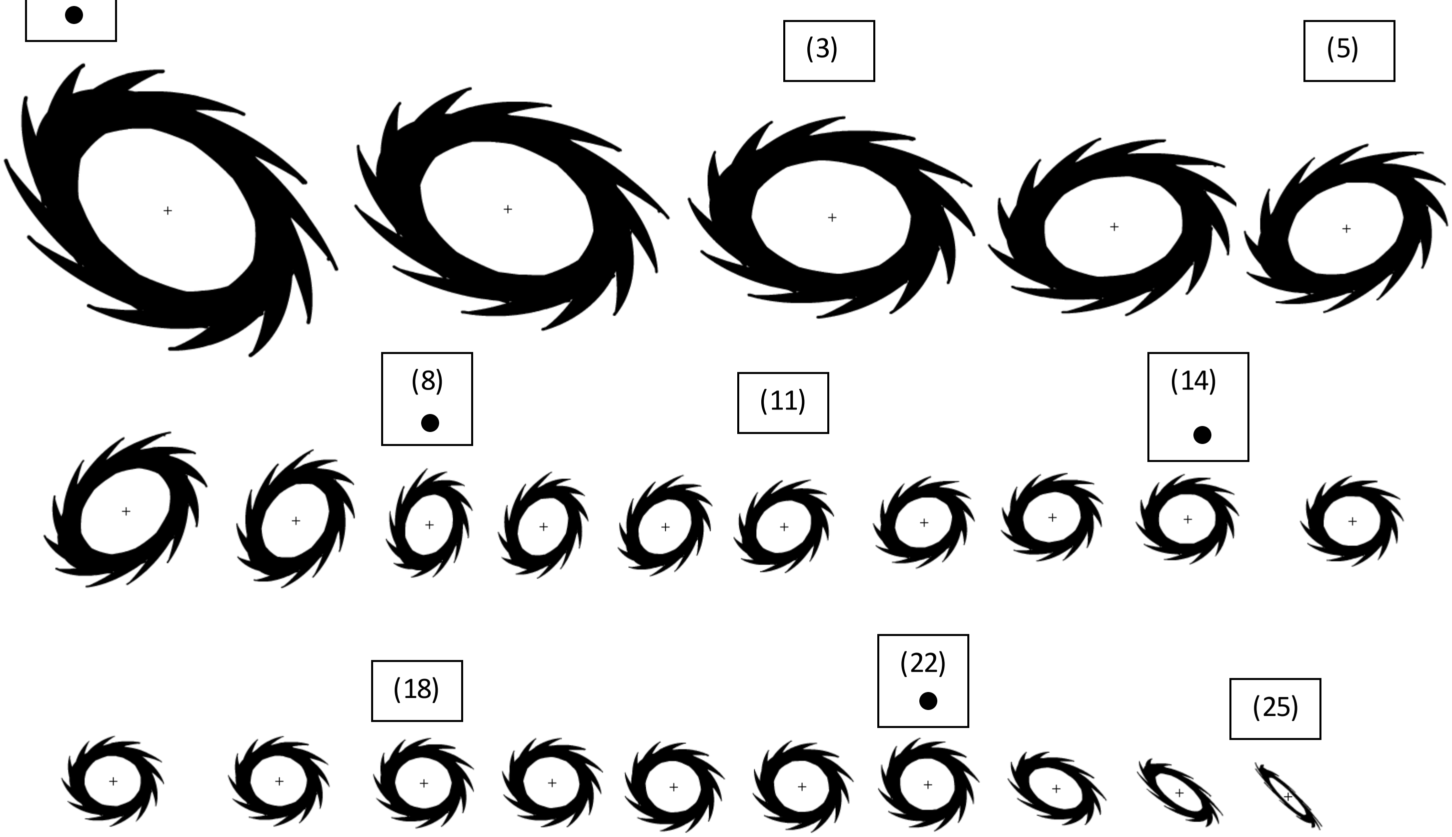


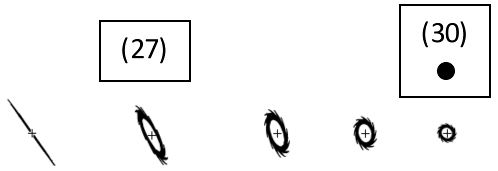
will

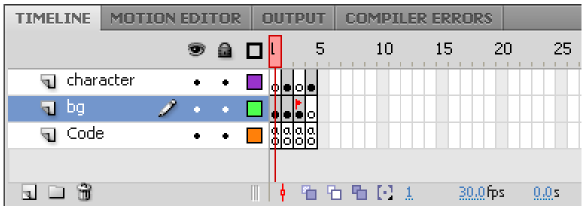
_root.nextFrame();

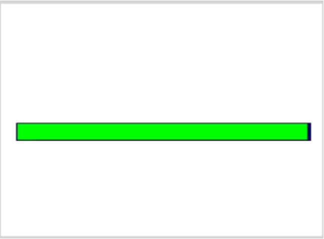


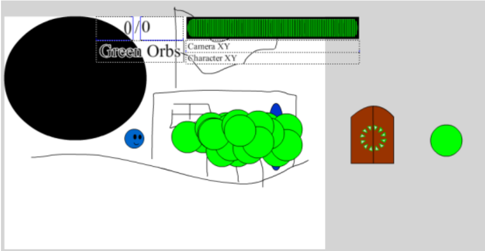
(1)

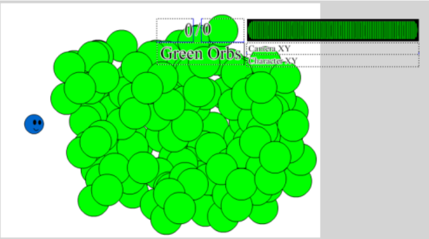








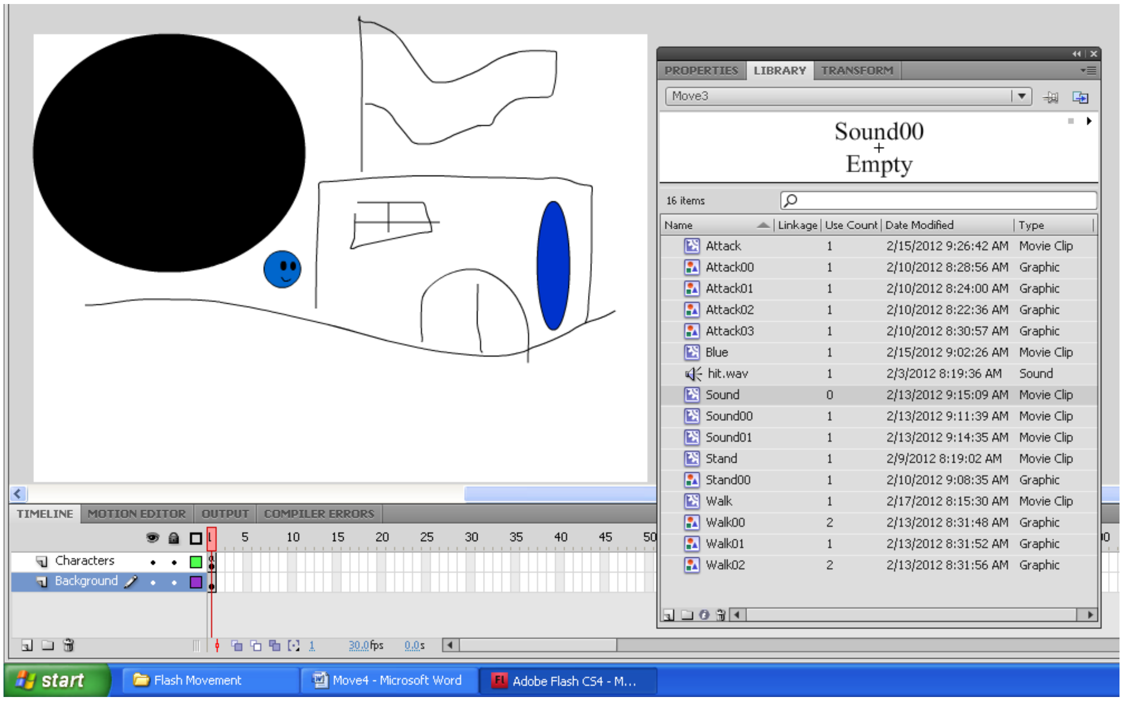




This Flash File runs at 30 frames per second.

When simulating a download, the loadbar appears after the file is about 44% loaded. If the hud's linkage is taken away, the loadbar shows up at 10% loaded.

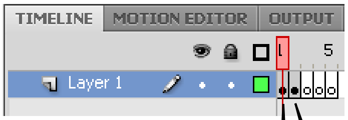
Most of the things in Move3 are in Move4 except a sound movie clip that has mysteriously disappeared…



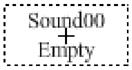
16 items

Name	Linkage	Use Count	Date Modified	Type
Attack		1	2/15/2012 9:26:42 AM	Movie Clip
Attack00		1	2/10/2012 8:28:56 AM	Graphic
Attack01		1	2/10/2012 8:24:00 AM	Graphic
Attack02		1	2/10/2012 8:22:36 AM	Graphic
Attack03		1	2/10/2012 8:30:57 AM	Graphic
Blue		1	2/15/2012 9:02:26 AM	Movie Clip
hit.wav		1	2/3/2012 8:19:36 AM	Sound
Sound		0	2/13/2012 9:15:09 AM	Movie Clip
Sound00		1	2/13/2012 9:11:39 AM	Movie Clip
Sound01		1	2/13/2012 9:14:35 AM	Movie Clip
Stand		1	2/9/2012 8:19:02 AM	Movie Clip
Stand00		1	2/10/2012 9:08:35 AM	Graphic
Walk		1	2/17/2012 8:15:30 AM	Movie Clip
Walk00		2	2/13/2012 8:31:48 AM	Graphic
Walk01		1	2/13/2012 8:31:52 AM	Graphic
Walk02		2	2/13/2012 8:31:56 AM	Graphic

Sound (MovieClip)

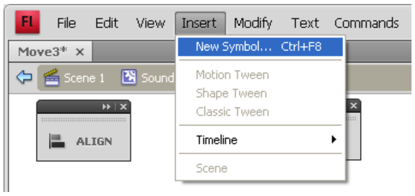


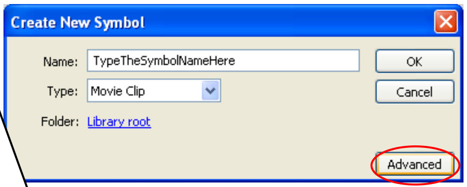
Sound00 (MovieClip)



This is the sound of me hitting a binder against the desk 4 times that I recorded with the Sound Recorder on my computer.

Adding linkage can make it possible to place things on the screen dynamically. This isn't used in Move3 or Move4, but here's how to add the Linkage:





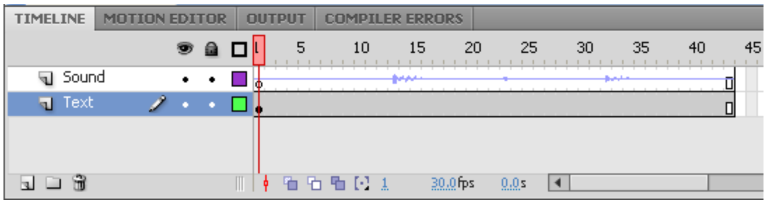
Create New Symbol

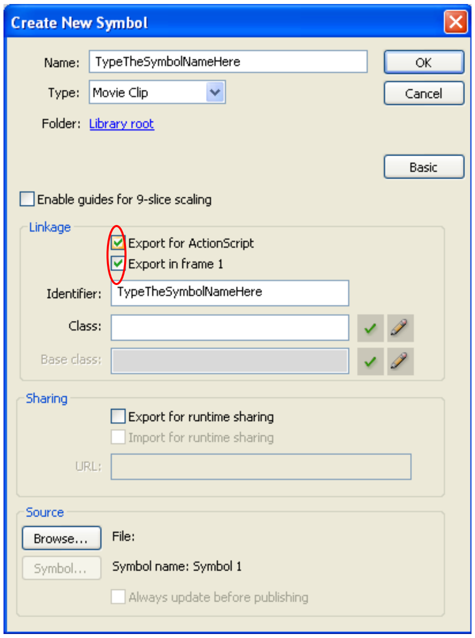
Name: TypeTheSymbolNameHere

Type: Movie Clip

Folder: Library root

Sound01 (MovieClip)





Create New Symbol

Name: TypeTheSymbolNameHere

Type: Movie Clip

Folder: Library root

☐ Enable guides for 9-slice scaling

Linkage

☑ Export for ActionScript

☑ Export in frame 1

Identifier: TypeTheSymbolNameHere

Class:

Base class:

Sharing

☐ Export for runtime sharing

☐ Import for runtime sharing

URL:

Source

File:

Symbol name: Symbol 1

☐ Always update before publishing