Cocos2d-X
-Action-

SoulSeek

~By : 스프라이트의 현재 속성값에 입력된 수치만큼 해당액션을 더한다. 즉, '여기서부터 세발자국 앞으로' 라는 표현처럼, 현재 산태에서 추가로 더해질 상태값을 지정.

~To : 입력된 수치까지 해당 액션을 수행한다. 즉, '저기로 가' 라는 표현처럼, 현재 위치나 속성과는 상관없이, 최종상태값을 지정.

Action

MoveBy와 MoveTo

MoveBy : 현재 위치에서 얼마나 더 이동할 것인지 정하는 액션.

MoveTo : 지정된 위치로 캐릭터를 무조건 이동.

```
//MoveTo(지속 시간, 크기)
auto pActionMTo = MoveTo::create(2.0f, Vec2(200, 0));
m_pMan->runAction(pActionMTo);

//MoveBy(지속 시간, 좌표)
auto pActionMBy = MoveBy::create(2.0f, Vec2(200, 0));
m_pWoman->runAction(pActionMBy);
```

JumpBy와 JumpTo

JumpBy : 지정된 시간동안, 현재 위치에서 지정된 위치값을 더해서 이동하면서, 지정된 높이만큼 점프하듯이 튕기면서 움직이는 액션.

JumpTo : 지정된 시간동안, 무조건 지정된 위치까지 이동하면서, 지정된 높이만큼 점프하는 듯이 튕기면서 액션.

```
//JumpTo(지속시간, 점프해서 이동되어지는 크기, 높이, 횟수 - 좌표이동하는 동안)
auto pActionJTo = JumpTo::create(1.0f, Vec2(200, 0) , 50 , 3);
m_pMan->runAction(pActionJTo);

//JumpBy(지속시간, 점프해서 이동될 좌표, 높이, 횟수 - 좌표 이동하는 동안)
auto pActionJBy = JumpBy::create(1.0f, Vec2(200, 0) , 50, 3);
m_pWoman->runAction(pActionJBy);
```

RotateBy, RotateTo

RotateBy : 현재의 각도 속성에 지정된 값만큼 추가회전한다. +(오른쪽), -(왼쪽) 값을 입력하면 회전방향이 변경

Rotate To : 현재 회전방향은 무시하고 입력된 회전값까지 스프라이트를 회전

```
//Rotate
//RotateTo(지속시간, 회전값)
auto pActionRTo = RotateTo::create(2.0f, 270);
m_pMan->runAction(pActionRTo);

//RotateBy(지속시간, 회전값)
auto pActionRBy = RotateBy::create(2.0f, 270);
m_pWoman->runAction(pActionRBy);
```

ScaleBy와 ScaleTo

ScaleBy : 현재크기에서 지정된 크기만큼 크기를 조절해준다.

ScaleTo : 스프라이트의 기본 크기로부터 지정된 크기까지 조절해 준다.

```
//ScaleTo(지속시간, 스케일 비율)
auto pActionSTo = ScaleTo::create(2.0f, 2);
m_pMan->runAction(pActionSTo);

//ScaleBy(지속시간, 스케일 비율)
auto pActionSBy = ScaleBy::create(2.0f, 2);
m_pWoman->runAction(pActionSBy);
```

Place & Blink

Place : 스프라이트의 위치를 지정하는 액션, 스프라이트 속성을 바꾸는 setposition과 기능은 거의 같지만 Place 액션은 다른 액션과 혼합해서 사용할때 유용하다.

Blink : 지정된 시간만큼 객체를 보였다가 숨겼다가, 즉 깜빡이게 하는 액션

```
//Place(목표위치) - 순간이동
auto pActionPlace = Place::create(Vec2(300, 200));
m_pMan->runAction(pActionPlace);

//Blink(지속시간, 횟수)
auto pActionBlink = Blink::create(2.0f, 5);
m_pWoman->runAction(pActionBlink);
```

Hide & Show

Hide : 화면에 스프라이트를 안보이게 할때 사용.

Show : 화면에 스프라이트를 보여줄때 사용.

```
//Hide, Show
if (m_pMan->isVisible())
{
    auto pActionHide = Hide::create();
    m_pMan->runAction(pActionHide);
}
else
{
    auto pActionShow = Show::create();
    m_pMan->runAction(pActionShow);
}
```

FadeIn, FadeOut

```
//FadeIn, FadeOut(지속시간)
if (m_pMan->getOpacity() == 0)
{
    auto pActionFadeIn = FadeIn::create(1.5f);
    m_pMan->runAction(pActionFadeIn);
}
else
{
    auto pActionFadeOut = FadeOut::create(1.5f);
    m_pMan->runAction(pActionFadeOut);
}
```

Remove함수

removeChild : 인자값으로 들어오는 해당 Sprite나 Layer 삭제

removeFromParentAndCleanup : 부모객체를 호출해서 나자신을 포함한 하위Layer나 Sprite를 삭제

removeSelf : 자기 자신을 삭제

```
//remove 함수
this->removeChild(m_pMan, true);
m_pMan->removeFromParentAndCleanup(true);
```

Reverse

- 지정된 액션을 반대로 실행하는 액션. 모든 액션에 대해 거꾸로 진행이 가능.

```
//reverse
auto pActionMBy = MoveBy::create(2.0f, Vec2(200, 50));
auto pActionJBy = JumpBy::create(1.0f, Vec2(200, 0), 50, 3);
auto pActionRe = pActionMBy->reverse();
m_pMan->runAction(pActionRe);
```

Sequence

- 여러개의 개별 액션들을 순서대로 진행될 수 있도록 하는 액션.

```
//Sequence(액션, 액션, 액션,,,,,nullptr)
auto pActionMBy = MoveBy::create(2.0f, Vec2(200, 0));
auto pActionRBy = RotateBy::create(2.0f, 270);

auto pActionSe = Sequence::create(pActionMBy, pActionRBy, nullptr);

m_pMan->runAction(pActionSe);
```

DelayTime

- squence 액션을 사용할때, 다른 액션이 진행되다가 잠깐 몇초동안 아무일 없이 대기할 필요가 있을때 사용.

```
//DelayTime(대기시간)
auto pActionMBy = MoveBy::create(1.0f, Vec2(200, 0));
auto pActionRe = pActionMBy->reverse();

auto pActionDelTime = DelayTime::create(0.5f);

auto pActionSe = Sequence::create(pActionMBy, pActionDelTime, pActionRe, NULL);
m_pMan->runAction(pActionSe);
```

Repeat, RepeatForever

Repeat : 지정된 액션을 여러번에 걸쳐서 반복해야하는 상황에 지정한 횟수 만큼 반복

Repeat : 지정된 액션을 무제한으로 반복하는 것.

```
//Repeat(액션, 반복횟수), RepeatForever(액션)
auto pActionMBy = MoveBy::create(2.0f, Vec2(200, 0));
auto pActionRe = pActionMBy->reverse();
auto pActionSe = Sequence::create(pActionMBy, pActionRe, nullptr);

auto pActionRep = Repeat::create(pActionSe, 3);

m_pMan->runAction(pActionRep);
```

BezierCurve

```
//BezierCurve
//임시위치조정
m_pMan->setPosition(Vec2(200, 230));

//베지어 선언
ccBezierConfig bezier;
//제어점1
bezier.controlPoint_1 = Vec2(50, 150);
//제어점2
bezier.controlPoint_2 = Vec2(350, 70);
//종착점
bezier.endPosition = Vec2(200, 0);

//베지어액션
BezierTo* pActionBezTo = BezierTo::create(2.0f, bezier);
m_pMan->runAction(RepeatForever::create(pActionBezTo));
```

CallbackAction & RemoveSelf

```
//Callback Action

m_pMan->setVisible(false);

auto pActionPlace = Place::create(Vec2(200, 200));
auto pActionDelTime = DelayTime::create(1.5f);
auto pActionShow = Show::create();
auto pActionMBy = MoveBy::create(2.0f, Vec2(200, 50));
auto pActionRe = pActionMBy->reverse();

auto callbackFunc = CallFuncN::create(CC_CALLBACK_1(ActionScene::callback, this));

auto pActionSe = Sequence::create(pActionPlace, pActionDelTime, pActionShow, callbackFunc
    , pActionMBy, nullptr);

m_pMan->runAction(pActionSe);
```

CallBack함수선언

```
void callback(Ref* pSender);
```

CallBack함수 내용 정의

```
void ActionScene::callback(Ref* pSender)
{
    LabelTTF* pLabel = LabelTTF::create("Callback 1 called",
        "fonts/arial.ttf", 30, Size(300, 300));
    pLabel->setPosition(Vec2(240, 100));
    pLabel->setColor(Color3B(0, 255, 0));
    addChild(pLabel);
}
```

(int)0

RemoveSelf

```
auto pActionSe = Sequence::create(pActionPlace, pActionDelTime, pActionShow, callbackFunc
    , pActionMBy, RemoveSelf::create(), nullptr);
```