# COCOS2D-X -CAPTER3-

SOULSEEK

# 1. ANIMATION

# 1. ANIMATION

- **스프라이트 애니메이션**
  - 여러장의 이미지를 등록하고 애니메이션 사용.
- **Atlas(SpriteSheet)**
  - 여러장의 이미지를 묶어둔 묶음.
- **Atlas를 사용하는 애니메이션**
  - **Atlas**를 텍스쳐로 등록해서 원하는 크기만큼 그려서 애니메이션 사용
- **plist를 사용하는 애니메이션**
  - **plist(XML**파일**)**의 정보를 사용해서 애니메이션 사용.

# 1. ANIMATION

## 스프라이트 애니메이션

- 첫 프레임을 미리 보여주기 위해 스프라이트 한 장 그려주고 애니메이션 생성 및 설정

```
//Sprite 초기화 생성
Sprite* m_pMan = Sprite::create("ani/grossini_dance_01.png");
m_pMan->setPosition(Vec2(240, 160));
addChild(m_pMan);

//애니메이션 생성
auto pAnimation = Animation::create();
//애니메이션 프레임간 소요되는 시간(0.3f)
pAnimation->setDelayPerUnit(0.3f);
```

- 애니메이션에 등록할 이미지를 추가한다.

```
//애니메이션에 Sprite등록
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_01.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_02.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_03.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_04.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_05.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_06.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_07.png");
pAnimation->addSpriteFrameWithFile("ani/grossini_dance_08.png");
```

# 1. ANIMATION

- **애니메이션을 플레이 하기 위해 Animate에 등록한다.**

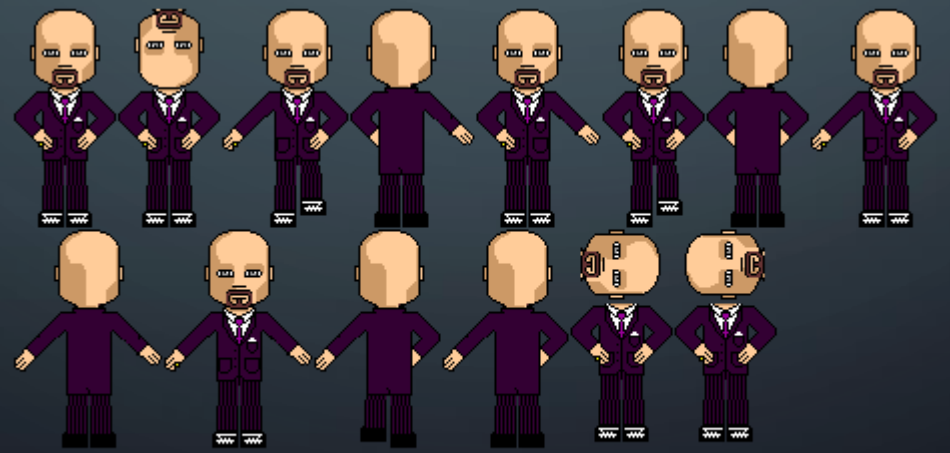
//애니메이트 생성
```
auto pAnimate = Animate::create(pAnimation);
m_pMan->runAction(pAnimate);
```

- **반복 플레이를 줘서 애니메이션을 계속 플레이 하게 한다**

//반복플레이를 줘서 반복 플레이를 하는 액션.
```
auto pAnimate = Animate::create(pAnimation);
auto pRefeatForever = RepeatForever::create(pAnimate);
m_pMan->runAction(pRefeatForever);
```

# 1. ANIMATION

- **Atlas**
- **Cocos2d - x**의 **Sprite**는 **OpenGL**의 특성상 크기보다 많이 읽는 것에 비용이 많이 든다**.**
- **Image**를 **Animation**에 사용하기 위해 한 장의 **Sheet**로 구성하는 것을 말한다**.**
- **Animation**용으로만 사용하는 것이 아니라 목적이 비슷하거나 같은 레이어 등등 하나의 **Atlas**로 묶어서 사용하면 된다**.**



# 1. ANIMATION

- **Atlas를 이용한 Animation**

- 텍스쳐를 등록한다. 두 가지 방법이 있다 하나만 택하면 된다.

- **Sprite**를 생성하고 **Texture**로 등록

```
auto m_pAniList = Sprite::create("grossini_.png");
auto pTexture = m_pAniList->getTexture();
```

- **Director**의 텍스쳐 캐시에 바로 등록

```
auto pTexture = Director::getInstance()->getTextureCache()->addImage("grossini_.png");
```

- 애니메이션을 생성하고 설정한다.
- 애니메이션 프레임에 등록한다.

```
auto pAnimation = Animation::create();
pAnimation->setDelayPerUnit(0.3f);

for (int i = 0; i < 14; i++)
{
	int colum = i % 5;
	int row = 1 / 5;

	//애니메이션에 등록(Atlas, x, y, width, height)
	pAnimation->addSpriteFrameWithTexture(pTexture, Rect(colum * 85, row * 121, 85, 121));
}
```

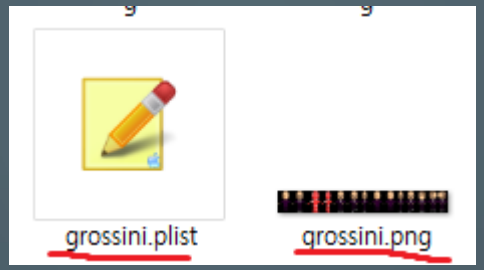
# 1. ANIMATION

- 첫 프레임을 보여준 스프라이트를 생성하고 애니메이션을 플레이 한다.

```
//스프라이트 생성 초기화
m_pMan = Sprite::createWithTexture(pTexture, Rect(0, 0, 85, 121));
m_pMan->setPosition(Vec2(240, 160));
addChild(m_pMan);

//애니메이션 실행
auto pAnimate = Animate::create(pAnimation);
auto pActionRep = RepeatForever::create(pAnimate);

m_pMan->runAction(pActionRep);
```

# 1. ANIMATION

- **plist를 사용하여 애니메이션하기 -** 같은 폴더상에 존재해야 한다.




- 캐시를 생성하고 스프라이트 프레임에 파일 등록한다.

```
SpriteFrameCache::getInstance()->addSpriteFramesWithFile("grossiniani.plist");
```

- **Vector**로 프레임을 생성하고 캐시에서 불러올 프레임명을 저장한 후 프레임에 저장.

```
Vector<SpriteFrame*> animFrames(16);
char szFileName[256] = { 0 };

for (int i = 1; i < 15; i++) // 1~7 , callback , 8 ~ 14
{
    //프레침명을 저장
    sprintf(szFileName, "grossini_dance_%02d.png", i);
    //프레임을 생성
    SpriteFrame* frame = SpriteFrameCache::getInstance()-
    >getSpriteFrameByName(szFileName);
    //애니메이션프레임에 등록
    animFrames.pushBack(frame);
}
```

# 1. ANIMATION

- plist의 내부를 볼 수 있다.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "ht
3   <plist version="1.0">
4   <dict>
5       <key>frames</key>
6       <dict>
7           <key>grossini_dance_01.png</key>
8           <dict>
9               <key>aliases</key>
10              <array/>
11              <key>spriteOffset</key>
12              <string>{0,-1}</string>
13              <key>spriteSize</key>
14              <string>{53,111}</string>
15              <key>spriteSourceSize</key>
16              <string>{85,121}</string>
```

# 1. ANIMATION

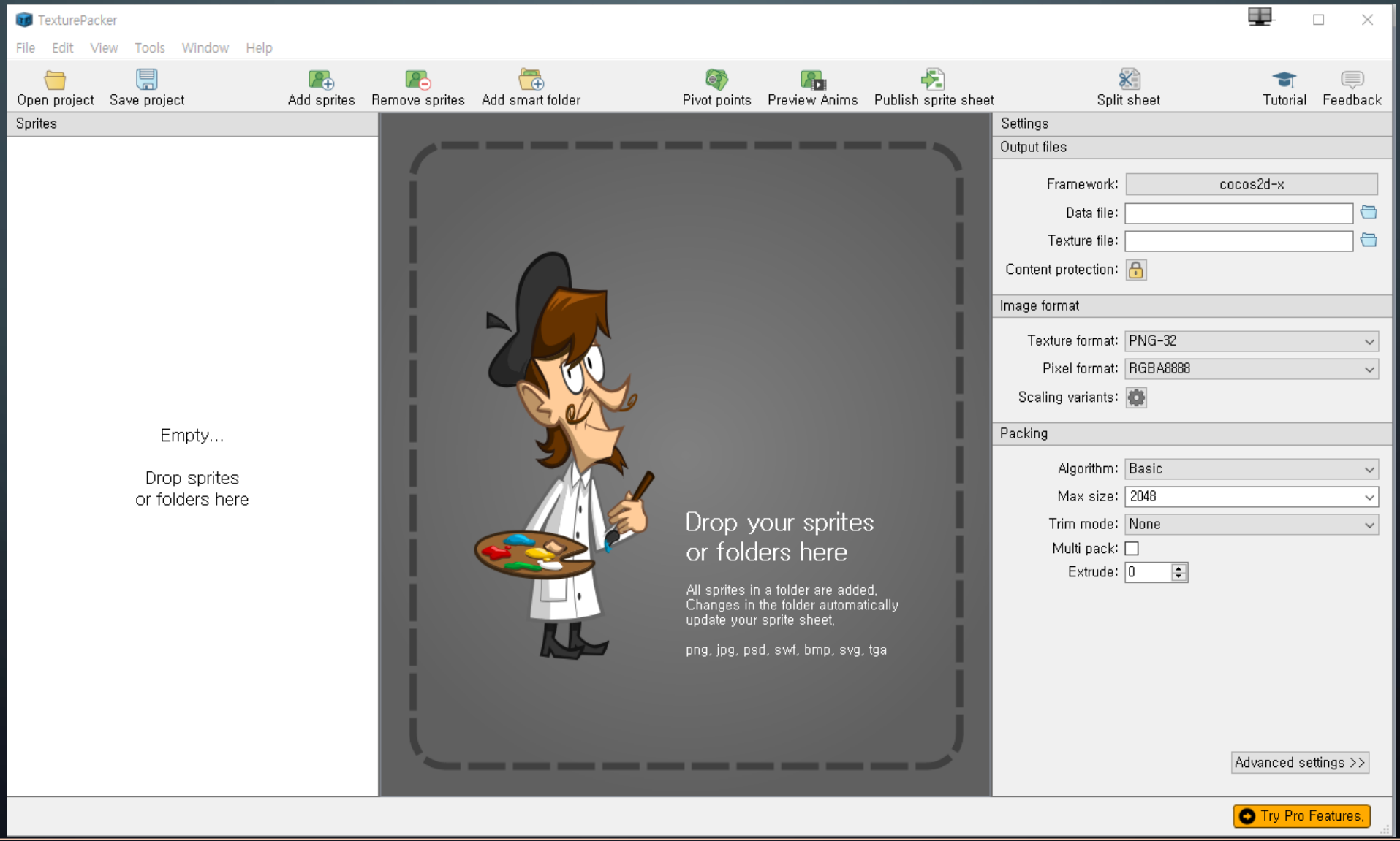
- 애니메이션을 프레임 **Vector**로 프레임을 생성하고 애니메이터에 등록한 후 플레이 한다.

```
//첫 번째 프레임 스프라이트를 생성한다.
auto pMan = Sprite::createWithSpriteFrameName("grossini_dance_01.png");
pMan->setPosition(Vec2(240, 160));
addChild(pMan);

//애니메이션 만들기
auto animation = Animation::createWithSpriteFrames(animFrames, 0.1f);
auto animate = Animate::create(animation);
auto actionrep = RepeatForever::create(pAnimate);

pMan->runAction(RepeatForever::create(animate));
```

# 1. ANIMATION

- pList를 생성하는 프로그램

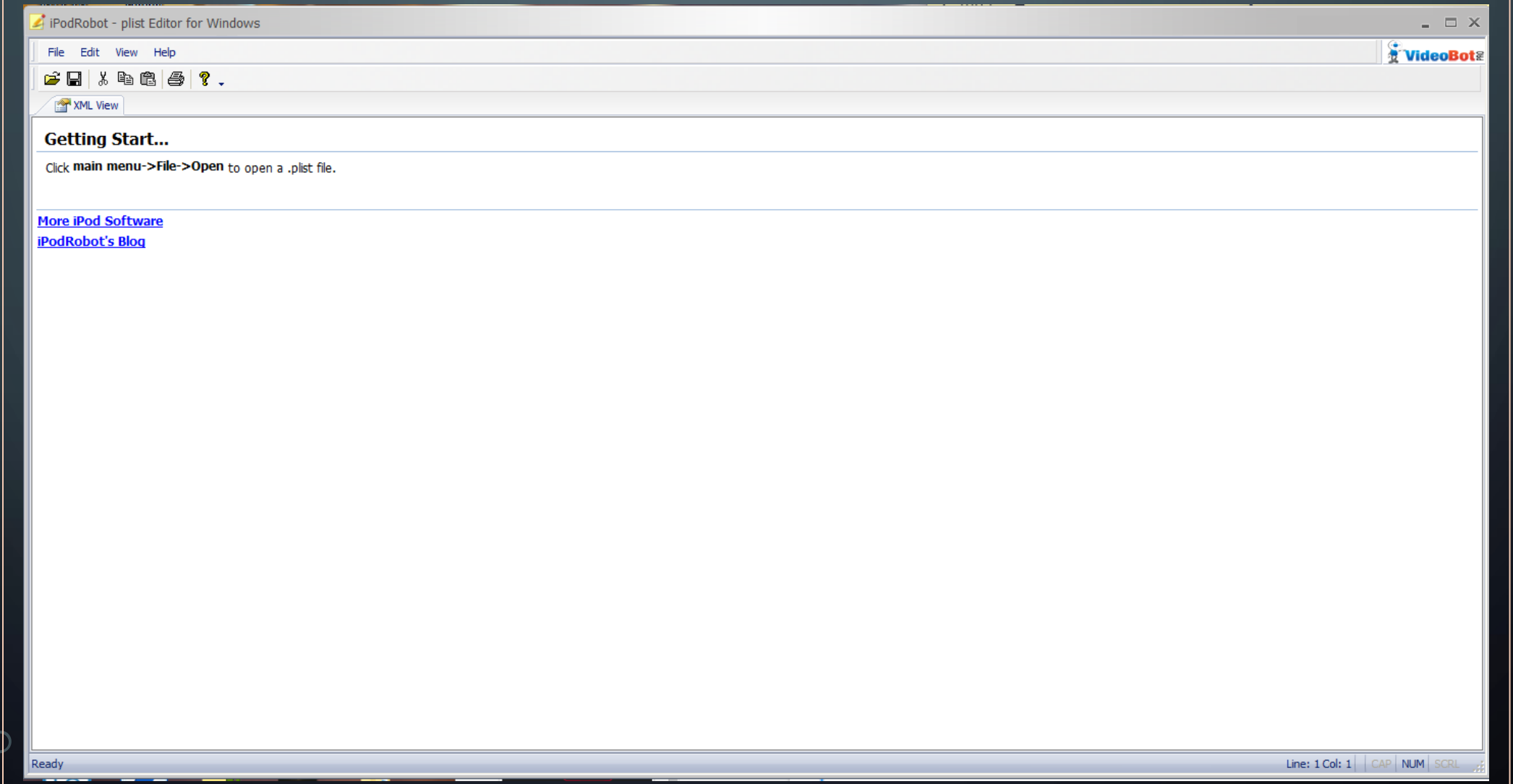
다운로드 : https://www.codeandweb.com/texturepacker



# 1. ANIMATION

- plist를 확인하는 프로그램

다운로드 : https://www.icopybot.com/download.htm



# 2. SCHEDULE

# 2. SCHEDULE

- **schedule(**스케쥴대상**) :** 매 프레임 마다 실행
- **schedule(**대상**,** 시간**) :** 지정된 시간마다 실행 하기
- **unschedule(), unscheduleUpdate()**로 멈추게 할수 있다**.**
- **scheduleUpdate() :** 제공하는 **update** 함수를 호출
- **update()** 함수를 만들어서 사용하자**.**

```
//매 프레임마다 실행
this->schedule(schedule_selector(InputEventScene::UserMove));

//매 초마다 실행
this->schedule(schedule_selector(InputEventScene::UserMove), 1.0f);

//제공된 업데이트 함수를 매초마다 호출
this->scheduleUpdate();

//schedule해제
this->unschedule(InputEventScene::UserMove);
this->unscheduleUpdate();
```

# 2. SCHEDULE

- **update** 함수
- 인자 값은 **float**형으로 고정이다**.**

```
void InputEventScene::update(float dt)
{
	if (m_bUp)
	{
		m_pWoman->setPosition(m_pWoman->getPosition() + Vec2(0, 2));
	}
}

void InputEventScene::UserMove(float f)
{
	m_pWoman->setPosition(m_pWoman->getPosition() + Vec2(0, 2));
}
```

*게임 전체를 멈추고 싶을 때 두 가지를 사용 한다

```
Director::getInstance()->pause();
Director::getInstance()->resume();
```