게임엔진

제15강 GUI

학습 안내

■학습 목표

□ 오우거 엔진의 2D 그래픽 요소의 출력 방법을 이해함으로써, 게임 UI 구현의 기본 지식을 습득한다.

■학습 내용

□ 오버레이를 이용한 문자 출력

□ 오버레이를 이용한 2D 이미지 출력 및 애니메이션

오버레이(Overlay)

■엔진의 메인 렌더링 위에 추가적으로 깔 수 있는 레이어.

■2D / 3D 요소를 추가할 수 있음.

■하드 코딩 및 오버레이(.overlay) 파일을 통해서 구현이 가능.

■오버레이 구조

- 한 개 이상의 Element 와 Container 블록의 조합으로 구성됨.
- Container 안에는 element 와 container 가 중첩될 수 있음.

■표준 오버레이 요소

- Panel - 사각형 패널, 컨테이너 또는 엘리먼트를 포함할 수 있음.
- BorderPanel - 가장자리를 꾸밀 수 있는 사각형 패널.
- TextArea - 텍스트 출력용

실습

TextOverlay
문자 출력 오버레이

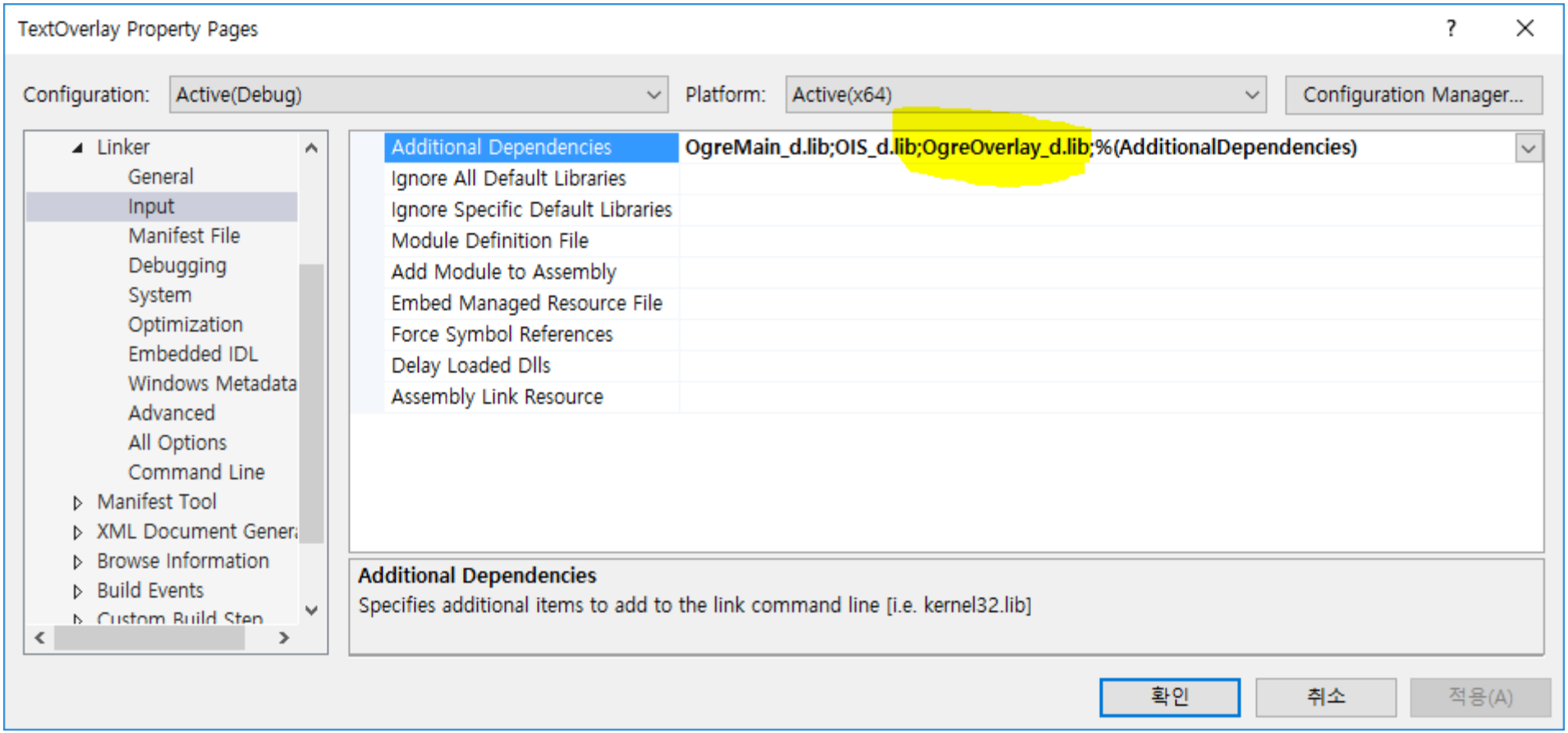
class LectureApp


```
void _setOverlay(void)
{
  mOverlayMgr = OverlayManager::getSingletonPtr();
  mTextOverlay = mOverlayMgr->create("TextOverlay");

  mPanel = static_cast<Ogre::OverlayContainer*>
    (mOverlayMgr->createOverlayElement("Panel", "container1"));
  mPanel->setDimensions(1, 1);
  mPanel->setPosition(-0.3f, 0.5f);

  OverlayElement* textBox = mOverlayMgr->createOverlayElement("TextArea", "TextID");
  textBox->setMetricsMode(Ogre::GMM_PIXELS);
  textBox->setPosition(10, 10);
  textBox->setWidth(100);
  textBox->setHeight(20);
  textBox->setParameter("font_name", "Font/NanumBold18");
  textBox->setParameter("char_height", "40");
  textBox->setColour(Ogre::ColourValue::White);
  textBox->setCaption(L"한국산업기술대학교 이대현 선수");
  mPanel->addChild(textBox);

  mTextOverlay->add2D(mPanel);
  mTextOverlay->show();
}
```

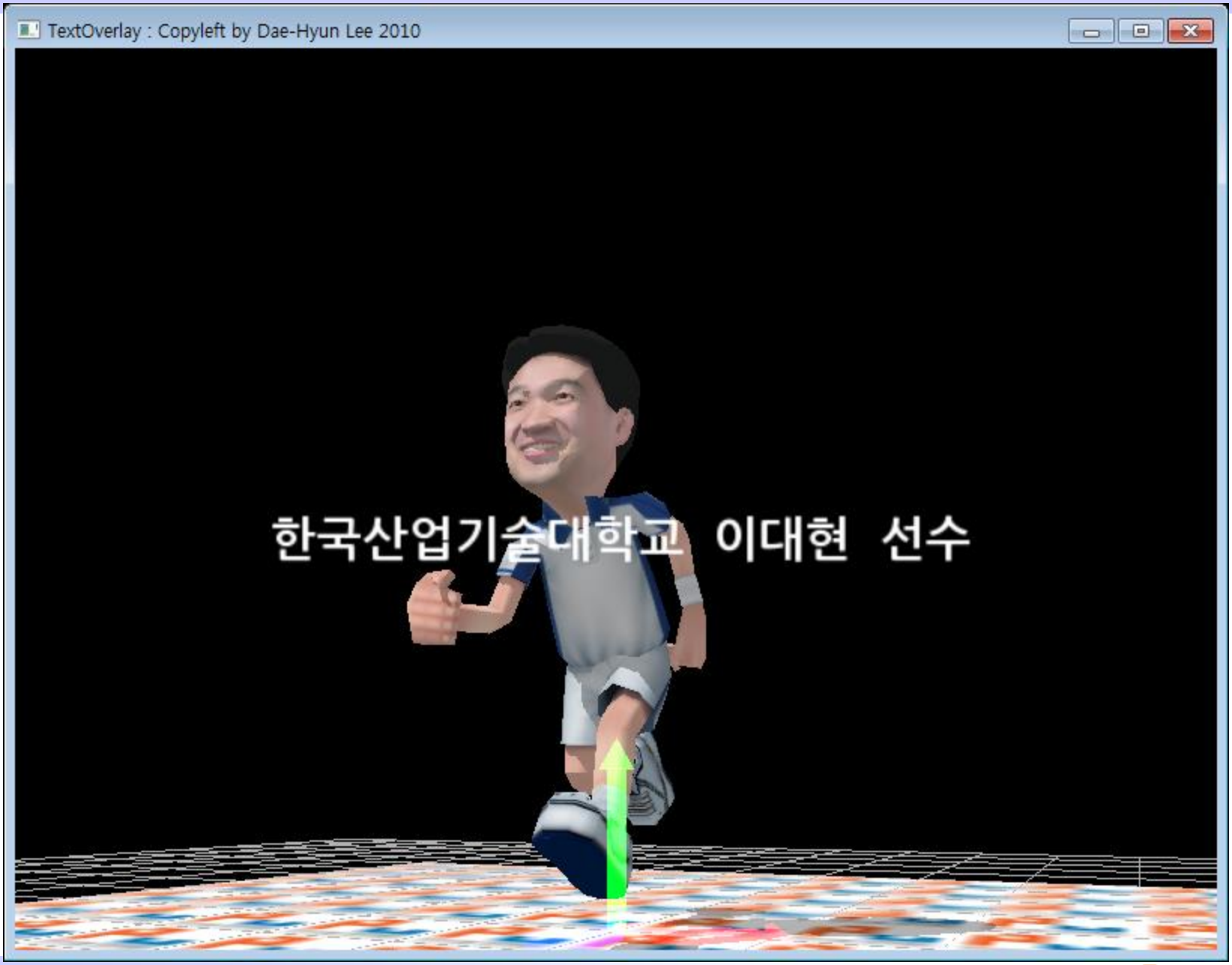
class InputController


```
bool frameStarted(const FrameEvent &evt)
{
  ... 중략 ...
  static float panelX = -0.3f;
  mPanel->setPosition(panelX, 0.5f);
  panelX = panelX > 1.0f ? -0.3f : panelX + 0.1f * evt.timeSinceLastFrame;
  ... 중략 ...

}
```

OgreOverlay 라이브러리 링크 - DLL 도 연결 필요



실행 결과



폰트의 정의: lecture.fontdef

■오우거 엔진의 폰트 처리 과정

1. 폰트매니저는 먼저 fontdef 파일안에서 정의된 내용을 바탕으로, 내부적으로 폰트를 렌더링 하여 커다란 2D 이미지를 생성한다.
2. 폰트매니저에게 글자의 유니코드가 넘어오면, 이미 만들어진 2D 이미지 내에서 유니코드에 해당되는 글자이미지를 찾아서 리턴시켜준다.

```
Font/NanumBold18
{
    type            truetype
    source          NanumBold.ttf
    size            18
    resolution      96
    code_points     33-167 4352-4607 12592-12687 44032-55203
}
```

엔진에서 사용되는 폰트 ID

폰트 이름 지정

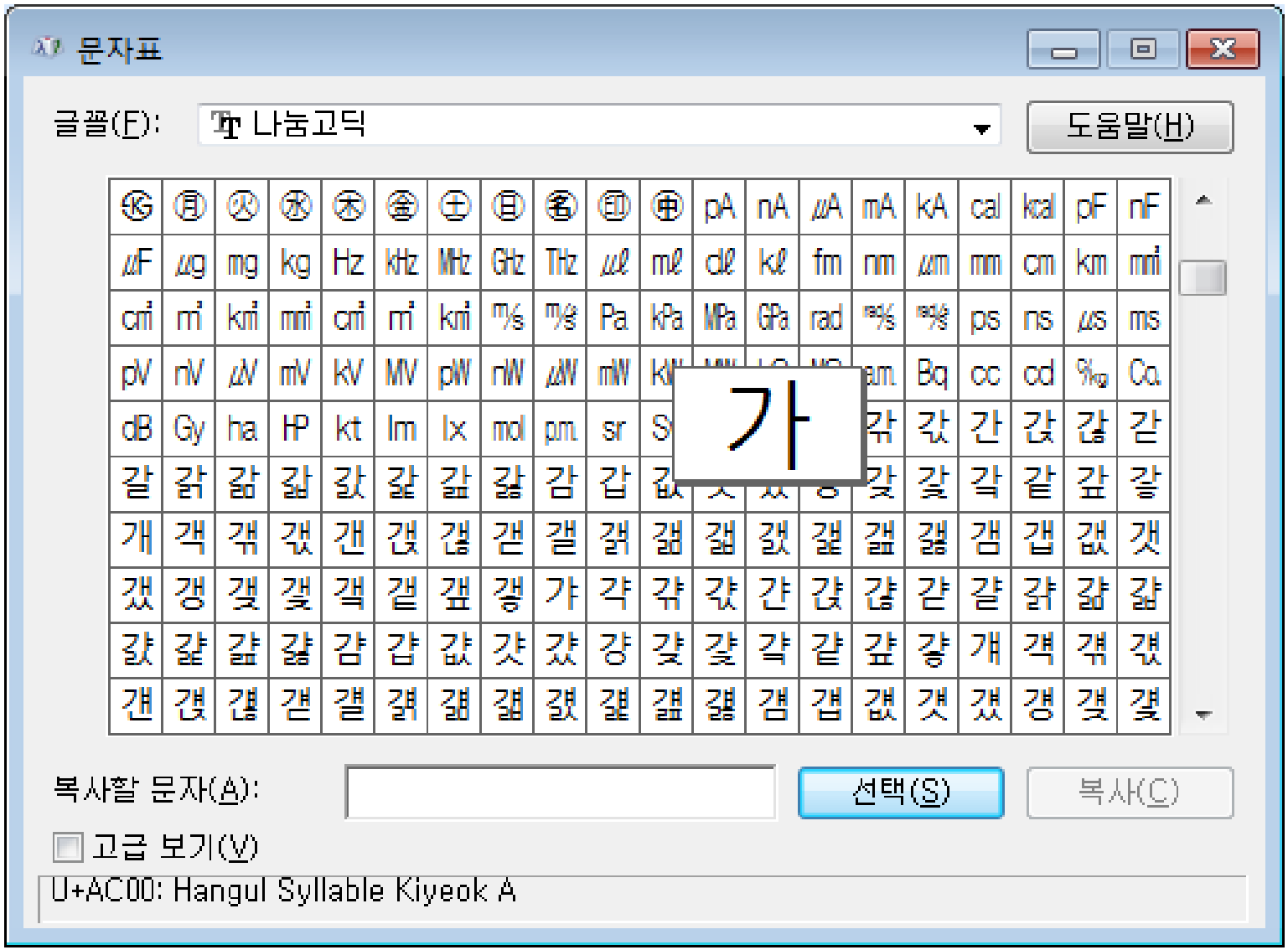
DPI

영문

유니코드 폰트에서 한국의 코드 범위를 지정함

11172자

폰트코드출력: charmap.exe



오버레이 시스템의 초기화

```
mOverlaySystem = new Ogre::OverlaySystem();


mSceneMgr->addRenderQueueListener(mOverlaySystem);
```

오버레이의 생성, 설정 및 표시

```
mOverlayMgr = OverlayManager::getSingletonPtr();
mTextOverlay = mOverlayMgr->create("TextOverlay");

mPanel = static_cast<Ogre::OverlayContainer*>
     (mOverlayMgr->createOverlayElement("Panel", "container1"));
mPanel->setDimensions(1, 1);

mPanel->setPosition(-0.3f, 0.5f);

mTextOverlay->show();
```

오버레이 생성

Container 타입

Container 이름

위치

오버레이 표시

화면 왼쪽 안쪽으로 30% 만큼 들어감

패널의 크기 (화면사이즈를 1로 했을때의 비율값)

텍스트 박스의 추가

```
OverlayElement* textBox = mOverlayMgr->createOverlayElement("TextArea", "TextID");


textBox->setMetricsMode(Ogre::GMM_PIXELS);

textBox->setPosition(10, 10);
textBox->setWidth(100);
textBox->setHeight(20);
textBox->setParameter("font_name", "Font/NanumBold18");
textBox->setParameter("char_height", "40");
textBox->setColour(Ogre::ColourValue::White);
textBox->setCaption(L"한국산업기술대학교 이대현 선수");


mPanel->addChild(textBox);

mTextOverlay->add2D(mPanel);
```

Element

픽셀 단위로 크기 지정 타입

모두 화면 Pixel 단위임 (상위 컨테이너에 대해 상대적)

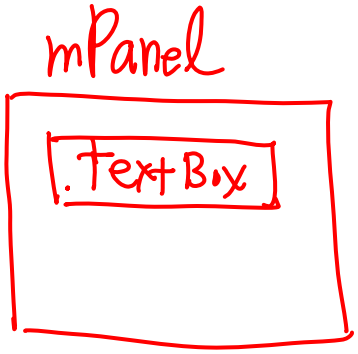
fontdef 에 정의된 폰트의 ID

글자의 색상

유니코드 String

컨테이너를 overlay에 할당

text Element를 패널에 삽입.

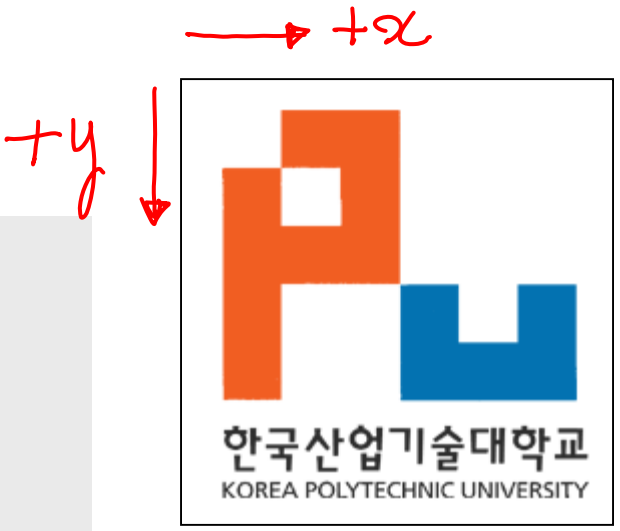


텍스트 박스의 이동

```
bool frameStarted(const FrameEvent &evt)
{

  static float panelX = -0.3f;
  mPanel->setPosition(panelX, 0.5f);
  panelX = panelX > 1.0f ? -0.3f : panelX + 0.1f * evt.timeSinceLastFrame;

}
```

패널의 위치를 update

바닥 텍스쳐 애니메이션




```
material KPU_LOGO
{
  technique
  {
    pass
    {
      texture_unit
      {
        scroll_anim 0 -4
        texture KPU_LOGO.gif
      }
    }
  }
}
```

x축 방향속도

y축 방향속도 −4 point(cm) / s

실습

ImageOverlay
2D 이미지 오버레이

class LectureApp


```
void _setOverlay(void)
{
  ... 중략 ...


  mLogoOverlay =
    OverlayManager::getSingleton().getByName("Overlay/KPU_LOGO");
  mLogoOverlay->show();
}
```

class InputController


```
bool frameStarted(const FrameEvent &evt)
{
  ... 중략 ...
  static float scale = 0.0f;
  mLogoOverlay->setScale(abs(cos(scale)), abs(cos(scale)));
  scale += 0.01f;
  ... 중략 ...

}
```

KPU_LOGO.overlay


```
Overlay/KPU_LOGO
{
    zorder 100
    container Panel(container2)
    {
        metrics_mode pixels
        horz_align right
        vert_align top
        top 0
        left -100
        width 100
        height 100
        material KPU_LOGO2
    }
}
```

실행 결과


ImageOverlay : Copyleft by Dae-Hyun Lee 2010
한국산업기술대학교 이대현 선수

오버레이 파일을 이용한 오버레이 처리

```
mLogoOverlay = OverlayManager::getSingleton().getByName("Overlay/KPU_LOGO");
```

리소스 파일에서 지정된 이름의 overlay를 가져옴

```
Overlay/KPU_LOGO
{
   zorder 100
   container Panel(container2)
   {
      metrics_mode pixels
      horz_align right
      vert_align top
      top 0
      left -100
      width 100
      height 100
      material KPU_LOGO2
   }
}
```

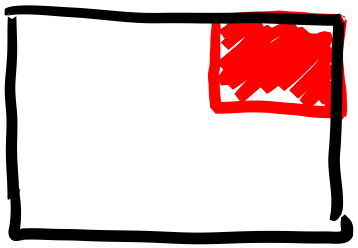
overlay ID

오버레이 순서 → 숫자가 클수록 위에 배치됨

pixel 단위 위치 지정

패널의 위치를 윈도우의 에 맞춤

왼쪽으로 지나 얼마큼 이동. 따라서



패널을 채울 이미지

오버레이 크기의 변경

```
bool frameStarted(const FrameEvent &evt)
{
  ... 중략 ...
  static float scale = 0.0f;
  mLogoOverlay->setScale(abs(cos(scale)), abs(cos(scale)));
  scale += 0.01f;
  ... 중략 ...

}
```

오버레이의 크기 조정.
단 원점의 위치도 변화에 따르므로 이를 따로 보정해주어야 함.

실습

InformationOverlay
프레임 속도 정보 출력 오버레이

class LectureApp


```
void _setOverlay(void)
{
  ... 중략 ...



  mInformationOverlay =
   OverlayManager::getSingleton().getByName("Overlay/Information");
  mInformationOverlay->show();
}
```

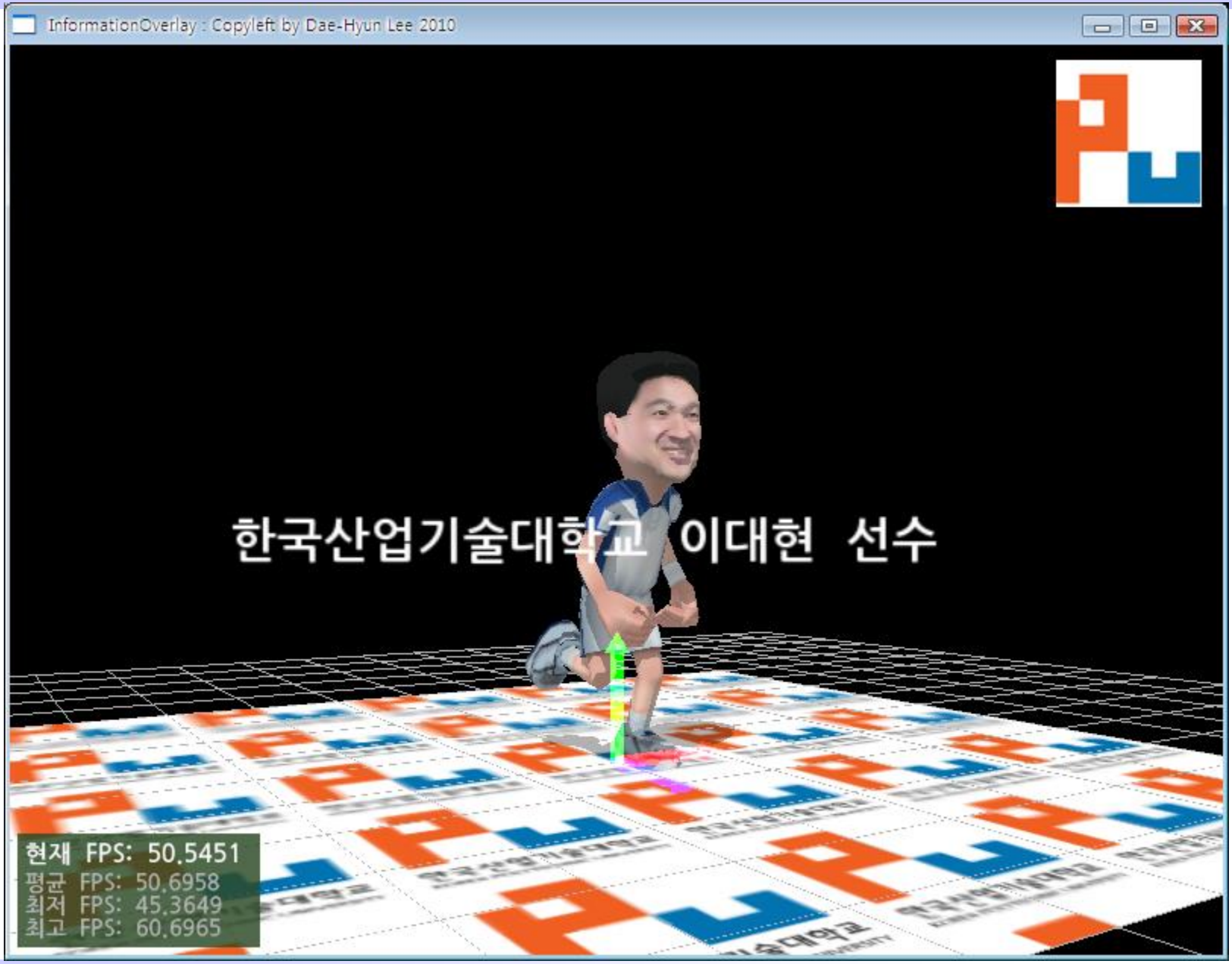
class InputController


```
bool frameEnded(const FrameEvent &evt)
{
  static Ogre::DisplayString currFps = L"현재 FPS: ";
  static Ogre::DisplayString avgFps = L"평균 FPS: ";
  static Ogre::DisplayString bestFps = L"최고 FPS: ";
  static Ogre::DisplayString worstFps = L"최저 FPS: ";
  OverlayElement* guiAvg =
    OverlayManager::getSingleton().getOverlayElement("AverageFps");
  OverlayElement* guiCurr =
    OverlayManager::getSingleton().getOverlayElement("CurrFps");
  OverlayElement* guiBest =
    OverlayManager::getSingleton().getOverlayElement("BestFps");
  OverlayElement* guiWorst =
    OverlayManager::getSingleton().getOverlayElement("WorstFps");
  const RenderTarget::FrameStats& stats =
    mRoot->getAutoCreatedWindow()->getStatistics();
  guiAvg->setCaption(avgFps + StringConverter::toString(stats.avgFPS));
  guiCurr->setCaption(currFps + StringConverter::toString(stats.lastFPS));
  guiBest->setCaption(bestFps + StringConverter::toString(stats.bestFPS));
  guiWorst->setCaption(worstFps + StringConverter::toString(stats.worstFPS));

  return mContinue;
}
```

오버레이 파일: Information.overlay


```
Overlay/Information
{
  zorder 500
  container BorderPanel(DebugPanel)
  {
  … 중략 …
     element TextArea(CurrFps)
        {
            metrics_mode pixels
            left 5
            top 5
            width 90
            height 30
            font_name Font/NanumBold18
            char_height 20
            caption Current FPS:
            colour_top 1 1 1
            colour_bottom 1 1 1
        }
     … 후략 …
```

실행 결과 - 프레임 속도 정보의 출력



프레임 속도 정보 출력


```
static Ogre::DisplayString currFps = L"현재 FPS: ";

OverlayElement* guiCurr = OverlayManager::getSingleton().getOverlayElement("
    CurrFps");
const RenderTarget::FrameStats& stats =
     mRoot->getAutoCreatedWindow()->getStatistics();

guiCurr->setCaption(currFps + StringConverter::toString(stats.lastFPS));
```

UTFString

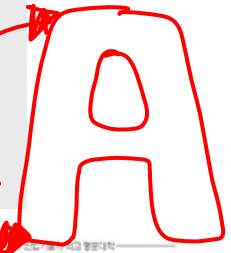
오버레이 요소 포인터 획득

각종 통계정보 획득

현재의 FPS

여기의 내용을 바꿈

```
element TextArea(CurrFps)
{
     metrics_mode pixels
     left 5
     top 5
     width 90
     height 30
     font_name Font/NanumBold18
     char_height 20
     caption Current FPS:
     colour_top 1 1 1
     colour_bottom 1 1 1
}
```



학습 정리

■오버레이

- □ 2D 그래픽 요소(텍스트 폰트 및 2D 이미지)의 출력에 활용.
- □ 구성 방법 - Container와 Element로 구성. 중첩 가능.
- □ 하드 코딩 또는 .overlay 파일을 통해서 오버레이 정보 제공.