게임엔진

제12강 게임 프레임웍

학습 안내

■학습 목표

- □ State Pattern 을 이용하여, 게임 프레임웍을 구성해본다.

■학습 내용

- □ 상태의 이해
- □ State Pattern 의 이해
- □ 스테이트 패턴을 이용한 게임 관리자 구현

Copyleft by 이대현

상태(State)

- A state is a condition in which an object can reside during its lifetime while it satisfies some condition, performs an activity, or waits for an event.

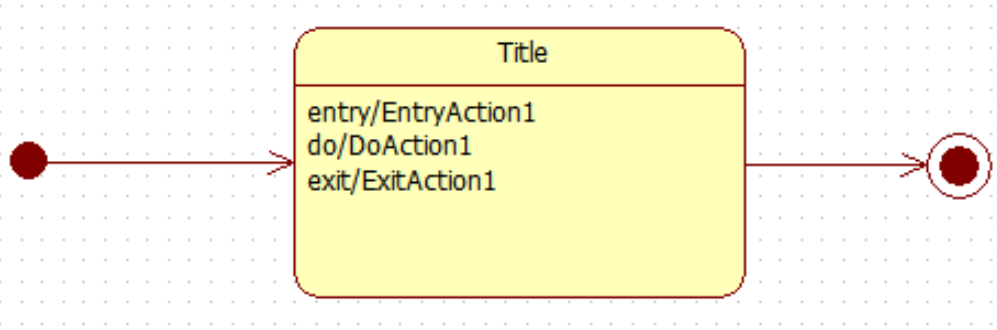
- 오브젝트가 놓여있는 상황으로써, 정해진 조건을 만족하는 동안, 요구되는 태스크를 지속적으로 수행하면서, 이벤트를 기다리며 머무르고 있게 됨.

- Entry Action - 오브젝트가 어떤 상태에 진입할 때, 처음으로 수행되는 일.

- Exit Action - 오브젝트가 어떤 상태를 빠져나갈 때, 마지막으로 수행되는 일.

- Event - 상태를 변경시키는 내/외부의 자극

- Do Activity - 오브젝트가 특정한 상태에 머무를 때, 수행하게 되는 일.



Copyleft by 이대현

Copyleft by 이대현

상태를 이용한 게임 구조 설계

■게임 상태란?

□ 게임 프로그램 실행 중의 어떤 특정 위치(또는 모드).

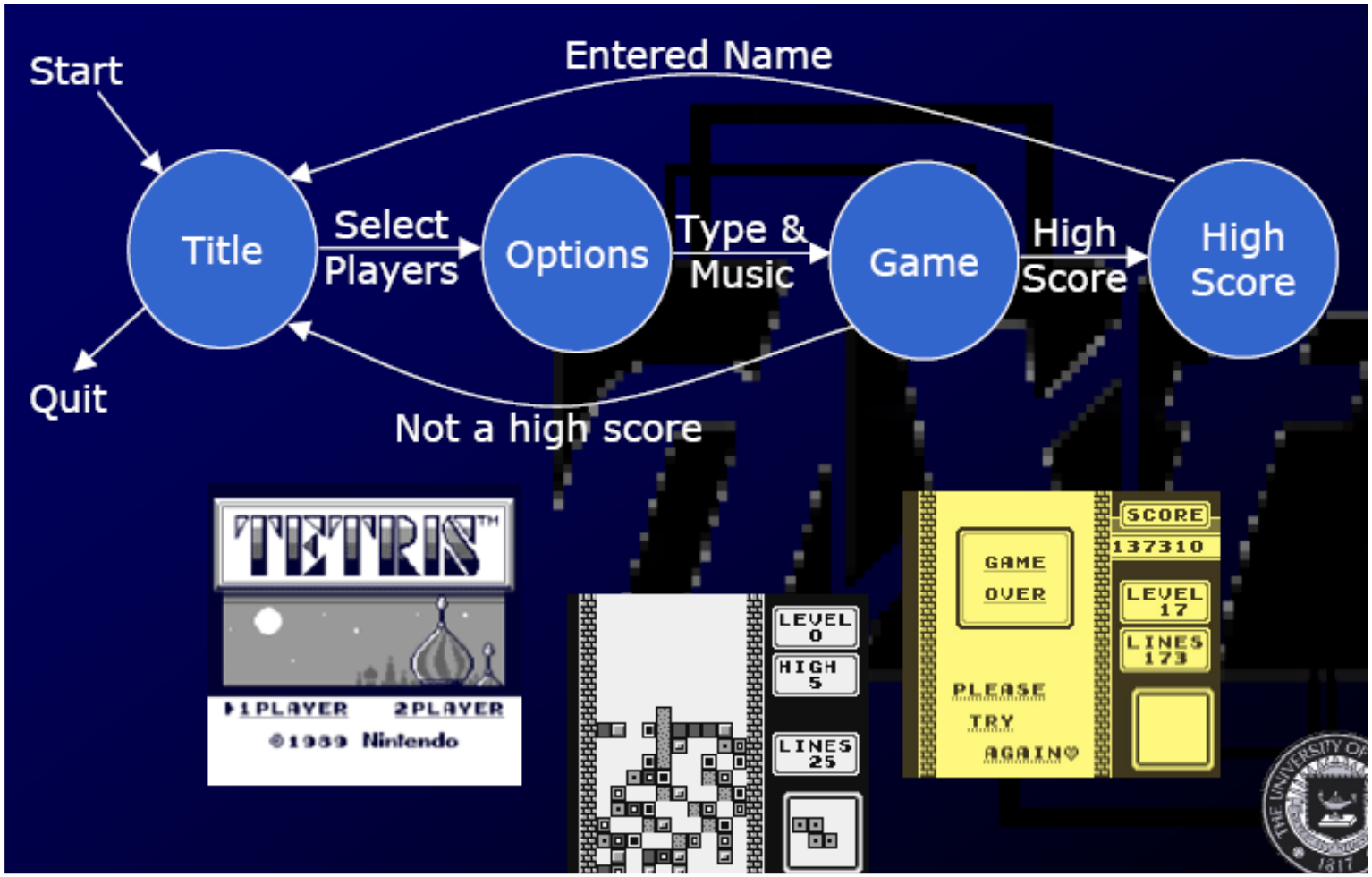
- 맵 선택 상태.
- 방향키는 맵 선택을 처리.

- 게임 메인 플레이 상태.
- 방향키는 캐릭터의 이동을 처리.

Copyleft by 이대현

■게임 프로그램은 게임 상태들의 집합으로 구성할 수 있음.

□ 예) 테트리스 게임



Copyleft by 이대현

■상태의 중첩이 가능함.

- □ 플레이 모드 내에서도 게임 상태의 세분화가 가능
- □ 가능하면 작은 단위의 게임 상태로 세분화할 수록 개발 및 디버깅이 용이.
  - ■ 예) 테니스 경기에서 서브 상태와 스트로크 상태의 구분.

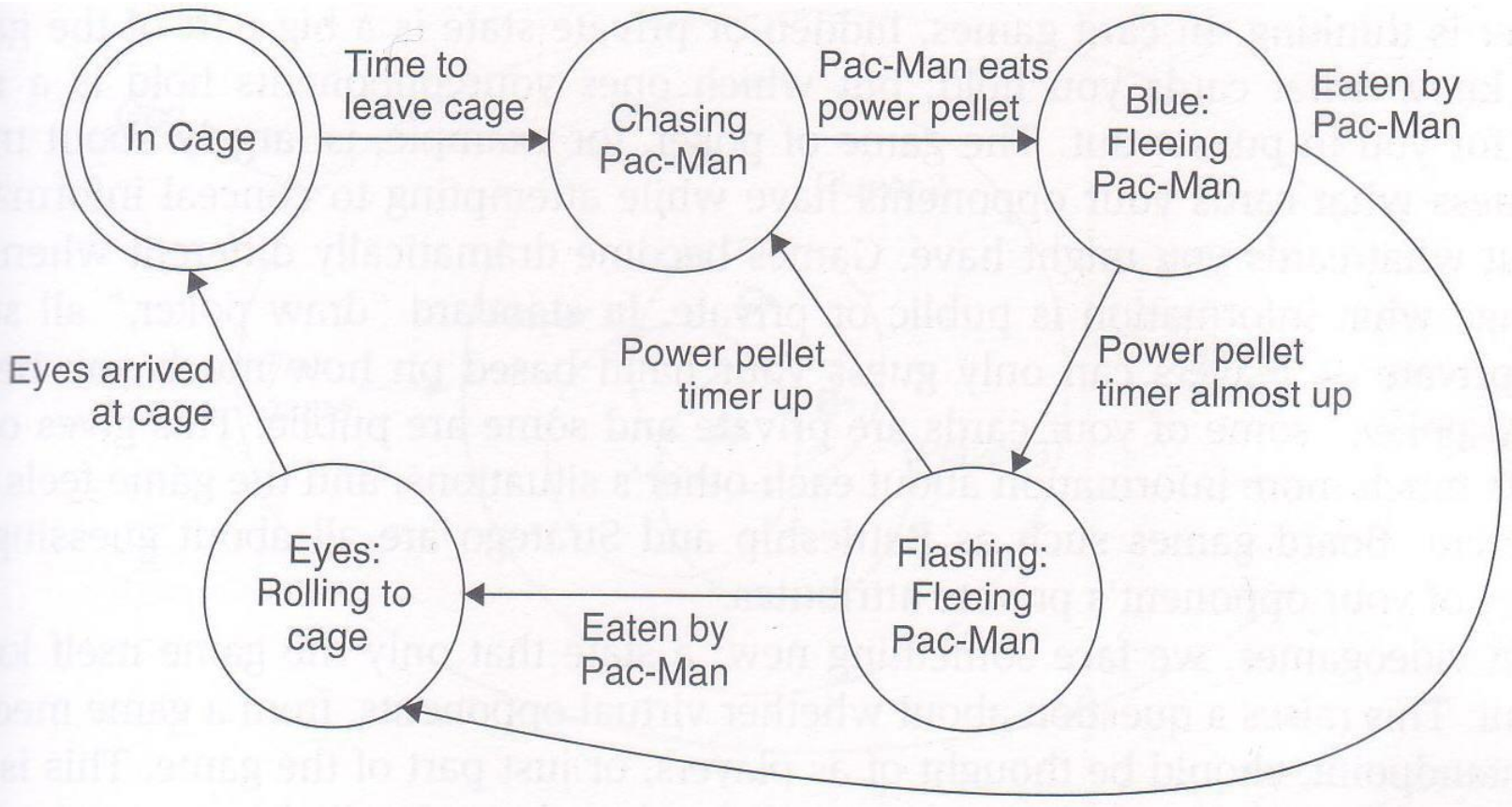
- 서브 상태.
- O 버튼은 서브 동작.

- 스크로크 상태.
- O 버튼은 스트로크 동작.

Copyleft by 이대현

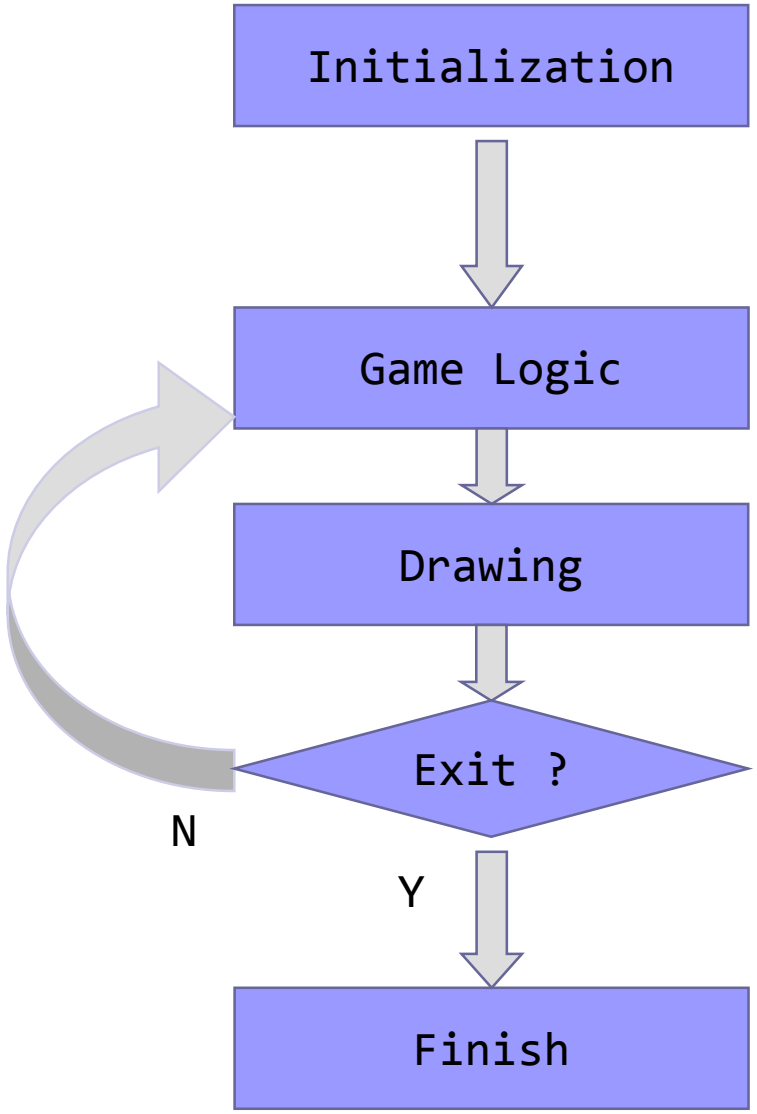
상태 다이어그램(State Diagram)

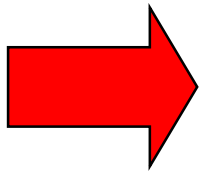
- 시스템의 변화를 모델링하는 다이어그램.
- 사건이나 시간에 따라 시스템 내의 객체들이 자신의 상태(state)를 바꾸는 과정을 모델링 함.

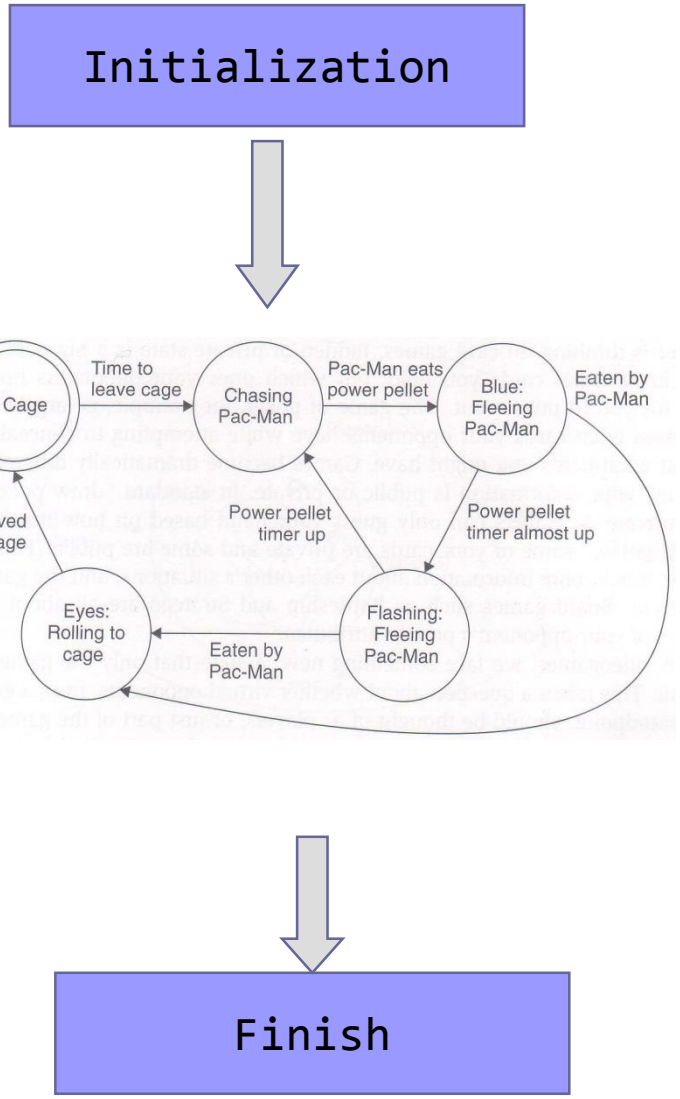


Copyleft by 이대현

게임 루프에서 게임 상태의 구현?

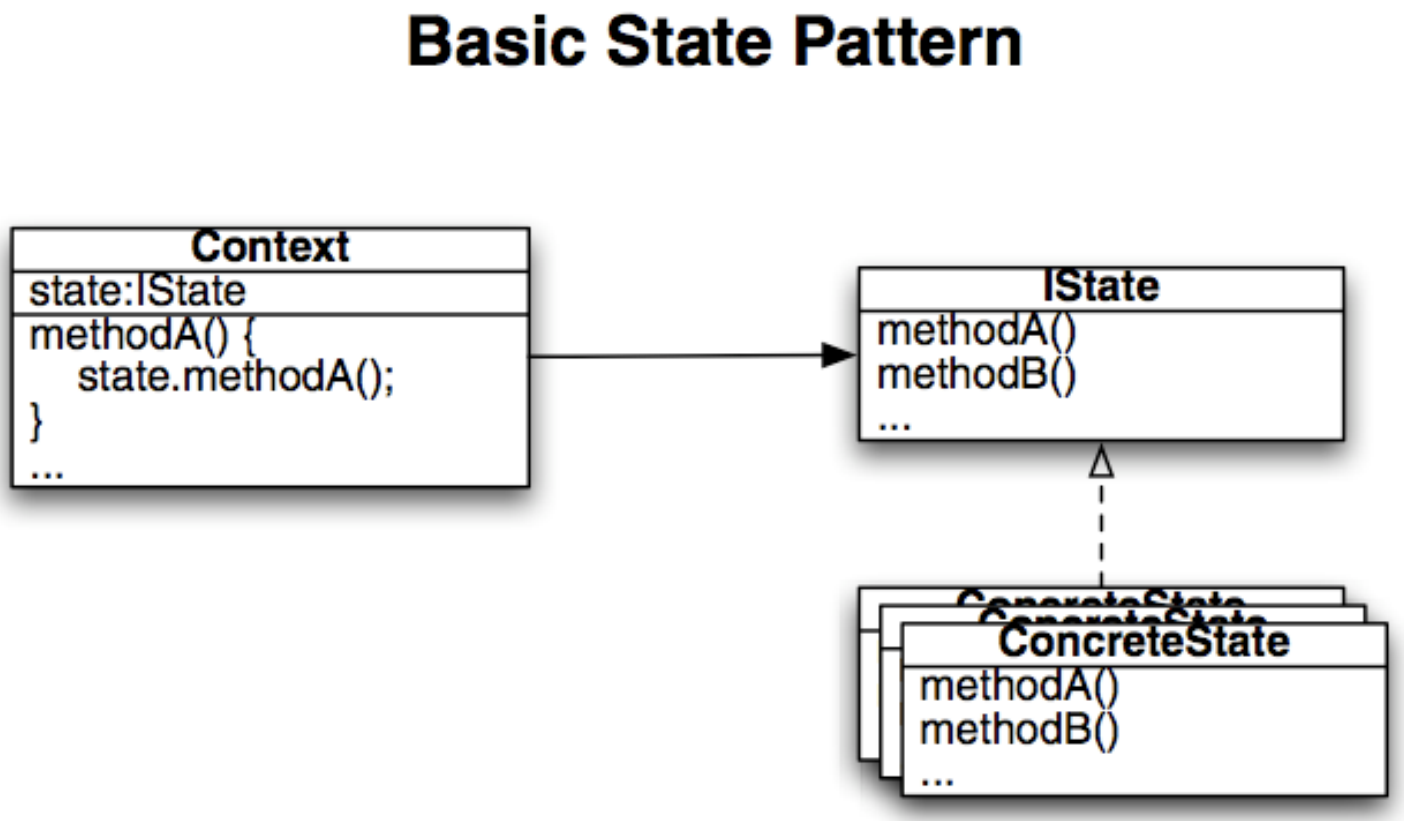






Copyleft by 이대현

스테이트 패턴(State Pattern)의 이용



Copyleft by 이대현

Copyleft by 이대현

프로젝트의 구성

■C++ 소스 파일들
- GameManager.cpp
- TitleState.cpp
- main.cpp
- PlayState.cpp

■C++ 헤더 파일들
- GameManager.h
- GameState.h
- TitleState.h
- PlayState.h

Copyleft by 이대현

main.cpp


```
… 전략 …

#if OGRE_PLATFORM == OGRE_PLATFORM_WIN32
  INT WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR strCmdLine, INT)
#else
  int main(int argc, char *argv[])
#endif
{
  GameManager game;
  try
  {
    game.init();
    game.changeState(TitleState::getInstance());
    game.go();

  }
  catch( Ogre::Exception& e )
  {

… 후략 …
```


Copyleft by 이대현

TitleState.h


```
class TitleState : public GameState
{
public:
  void enter();
  void exit();

  void pause();
  void resume();

  bool frameStarted(GameManager* game, const Ogre::FrameEvent& evt);
  bool frameEnded(GameManager* game, const Ogre::FrameEvent& evt);

  bool mouseMoved(GameManager* game, const OIS::MouseEvent &e)
  { return true; }

  bool mousePressed(GameManager* game, const OIS::MouseEvent &e, OIS::MouseButtonID id )
  { return true; }

  bool mouseReleased(GameManager* game, const OIS::MouseEvent &e, OIS::MouseButtonID id)
  { return true; }

  bool keyPressed(GameManager* game, const OIS::KeyEvent &e);
  bool keyReleased(GameManager* game, const OIS::KeyEvent &e) { return true; }

  static TitleState* getInstance() { return &mTitleState; }

  ... 후략 ...
```


Copyleft by 이대현

TitleState.cpp


```
bool TitleState::keyPressed(GameManager* game, const OIS::KeyEvent &e)
{
  switch(e.key)
  {
  case OIS::KC_SPACE:
    game->changeState(PlayState::getInstance());
    break;
  case OIS::KC_ESCAPE:
    mContinue = false;
    break;
  }

  return true;
}
```

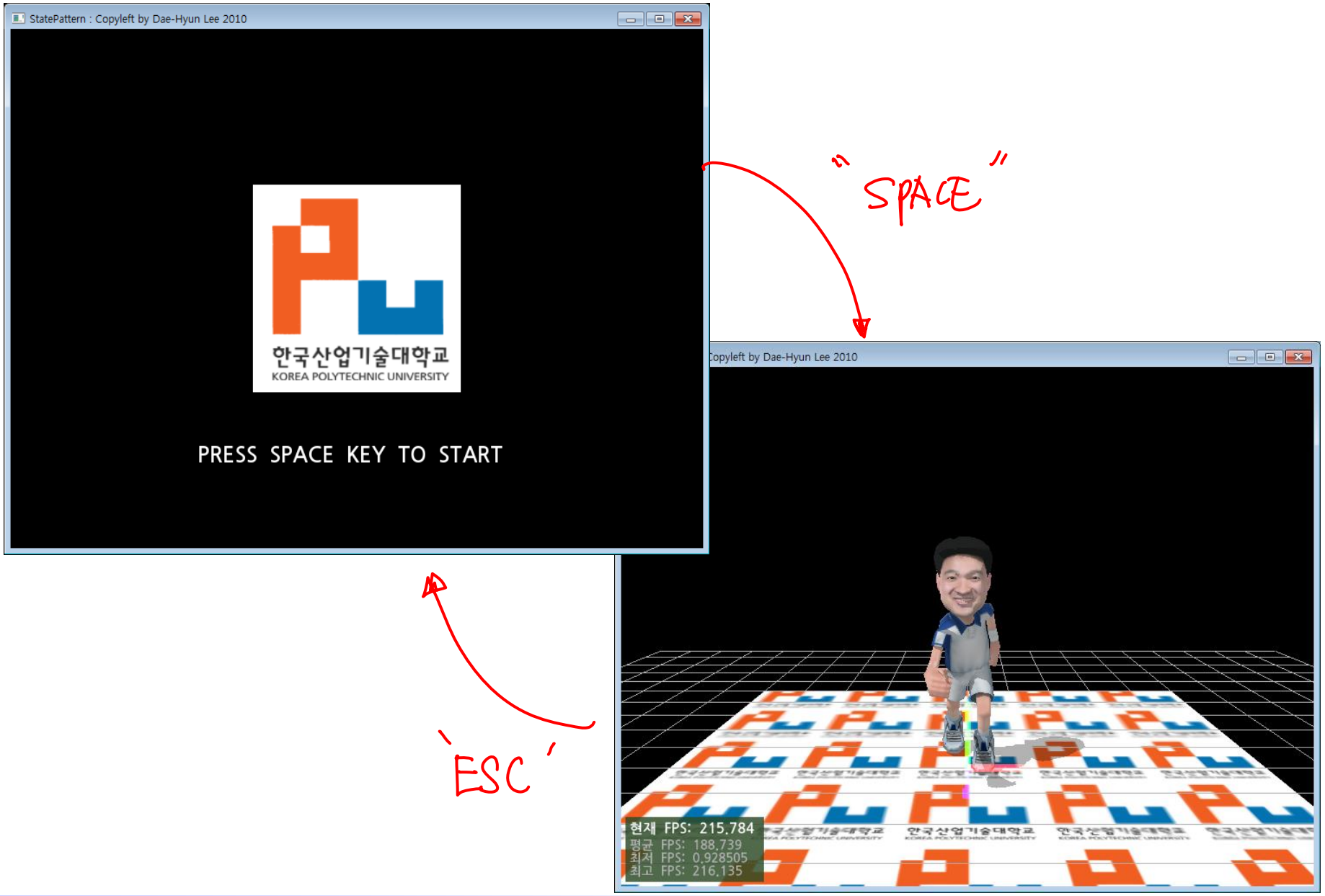

Copyleft by 이대현

PlayState.cpp


```
void PlayState::exit()
{
  mSceneMgr->clearScene();
  mInformationOverlay->hide();
}


bool PlayState::keyPressed(GameManager* game, const OIS::KeyEvent &e)
{
  switch(e.key)
  {
  case OIS::KC_ESCAPE:
    game->changeState(TitleState::getInstance());
    break;
  }

  return true;
}
```


Copyleft by 이대현

실행 결과

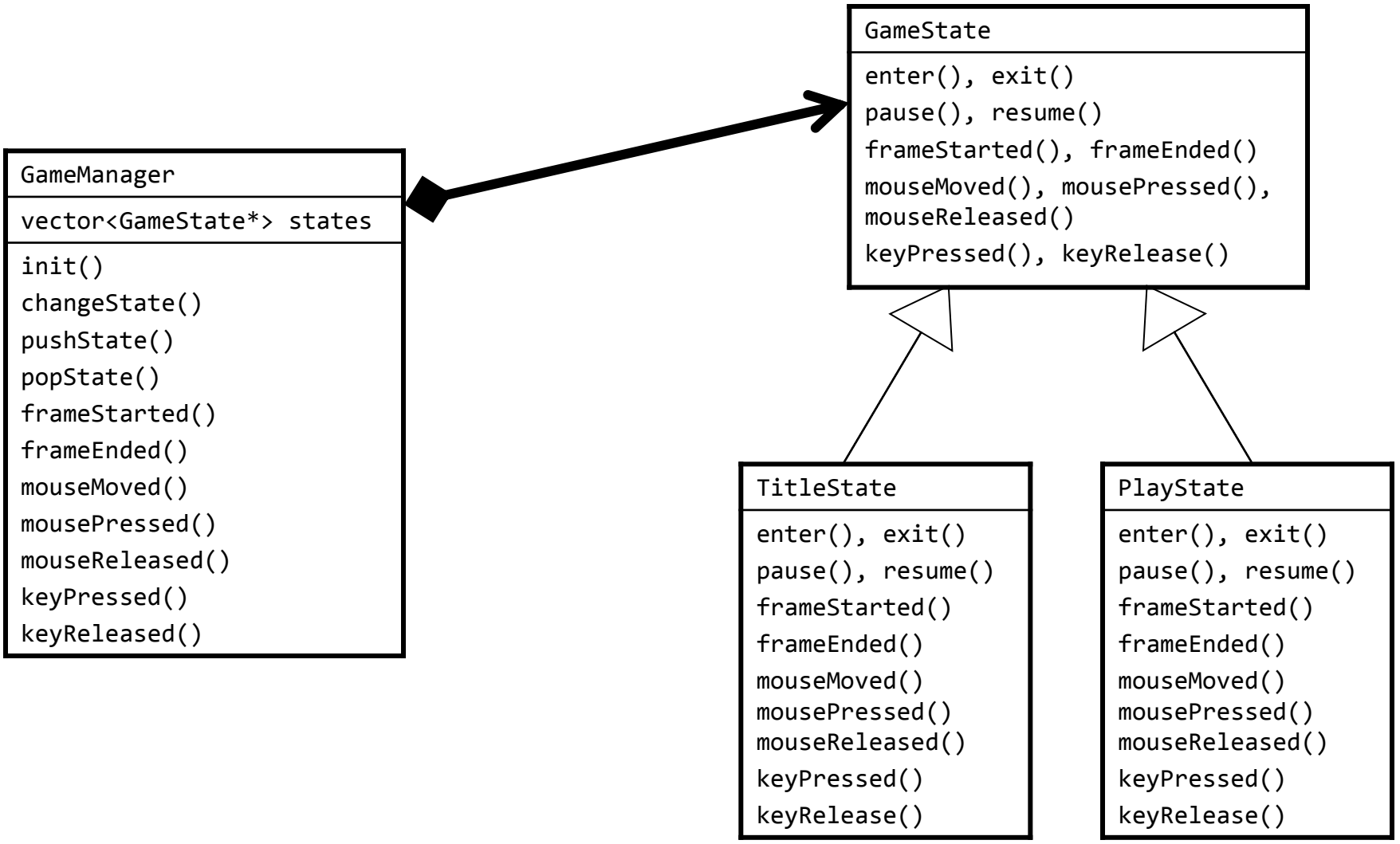

StatePattern : Copyleft by Dae-Hyun Lee 2010
한국산업기술대학교
KOREA POLYTECHNIC UNIVERSITY
PRESS SPACE KEY TO START
"SPACE"
Copyleft by Dae-Hyun Lee 2010
현재 FPS: 215.784
평균 FPS: 188.739
최저 FPS: 0.928505
최고 FPS: 216.135
`ESC'

게임 상태의 구현: GameState 클래스

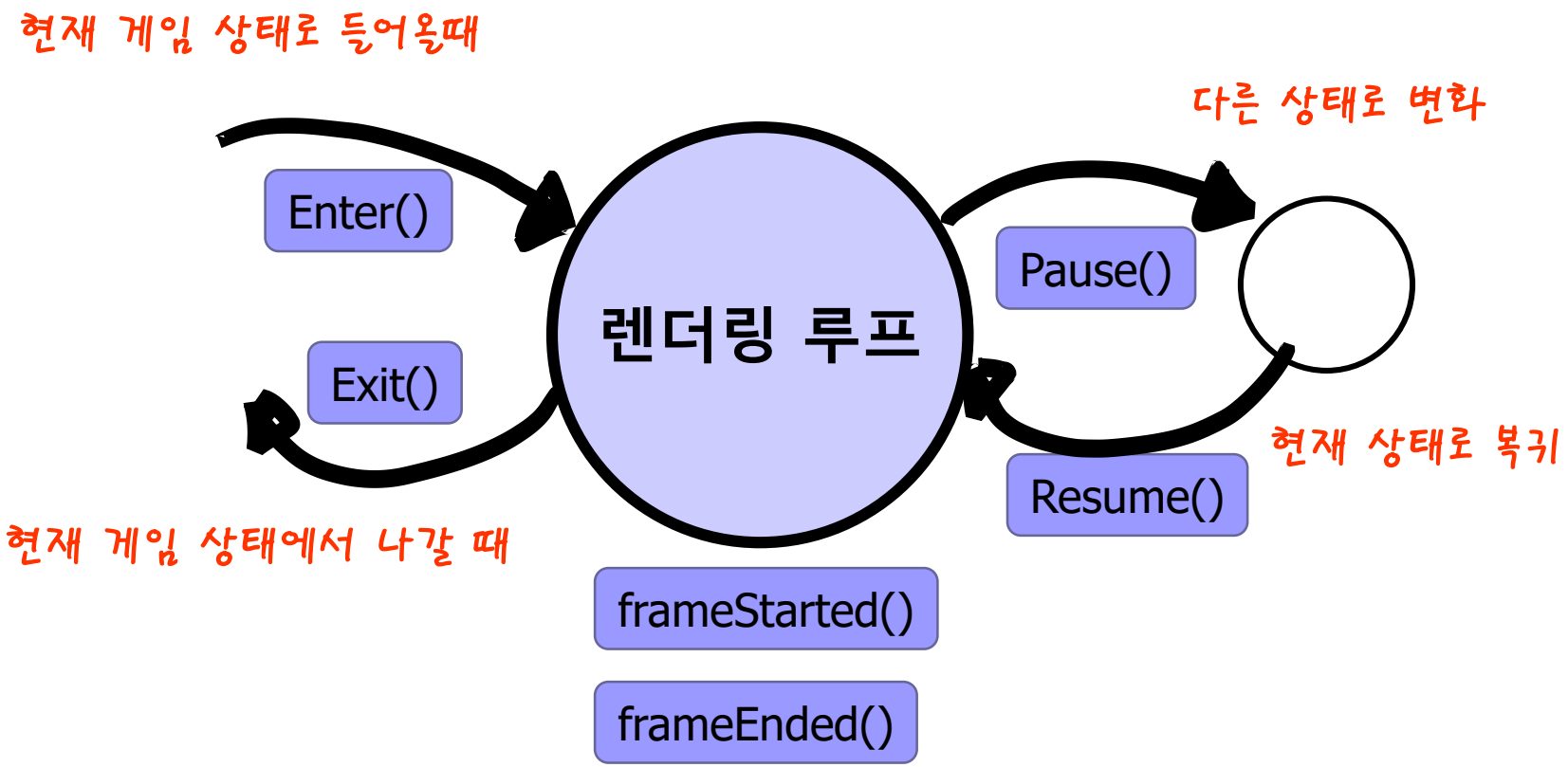
```
class GameState
{
public:
    virtual void enter(void) = 0;
    virtual void exit(void) = 0;

    virtual void pause(void) = 0;
    virtual void resume(void) = 0;

    virtual bool frameStarted(GameManager* game, const Ogre::FrameEvent& evt) = 0;
    virtual bool frameEnded(GameManager* game, const Ogre::FrameEvent& evt) = 0;

    virtual bool mouseMoved(GameManager* game, const OIS::MouseEvent &e) = 0;
    virtual bool mousePressed(GameManager* game, const OIS::MouseEvent &e, OIS::MouseButtonID id)=0;
    virtual bool mouseReleased(GameManager* game, const OIS::MouseEvent &e, OIS::MouseButtonID id)=0;

    virtual bool keyPressed(GameManager* game, const OIS::KeyEvent &e) = 0;
    virtual bool keyReleased(GameManager* game, const OIS::KeyEvent &e) = 0;

    void changeState(GameManager* game, GameState* state)
    {
        game->changeState(state);
    }

};
```


Copyleft by 이대현

State Pattern 구현



Copyleft by 이대현



Copyleft by 이대현

게임 관리자의 구현: GameManager 클래스

```
class GameManager :
  public Ogre::FrameListener,
  public OIS::KeyListener,
  public OIS::MouseListener
{
public:
  GameManager();
  ~GameManager();

  void init(void);
  void changeState(GameState* state);
  void pushState(GameState* state);
  void popState();

  void go(void);

  bool mouseMoved( const OIS::MouseEvent &e );
  bool mousePressed( const OIS::MouseEvent &e, OIS::MouseButtonID id );
  bool mouseReleased( const OIS::MouseEvent &e, OIS::MouseButtonID id );

  bool keyPressed( const OIS::KeyEvent &e );
  bool keyReleased( const OIS::KeyEvent &e );
```

- 게임 상태의 이동.
- 현재 상태를 소멸시킴.

- 게임 상태의 이동.
- 현재 상태를 메모리에 유지한 채로 다음 상태로 이동함.

- 이전 상태로 이동.
- 현재 상태는 소멸됨.

- 입력 처리

Copyleft by 이대현

게임 관리자를 프레임 리스너로 활용

```
void GameManager::init(void)
{
  ... 중략 ...

  mRoot->addFrameListener(this);
}

void GameManager::go(void)
{
  if (mRoot)
    mRoot->startRendering();
}

bool GameManager::frameStarted(const FrameEvent& evt)
{
  if(mMouse)
    mMouse->capture();
  if(mKeyboard)
    mKeyboard->capture();
  return states.back()->frameStarted(this, evt);
}

bool GameManager::frameEnded(const FrameEvent& evt)
{
  return states.back()->frameEnded(this, evt);
}
```

GameManager 를 프레임 리스너로 등록

GameManager 의 두 함수를 호출하게 됨.

GameManager 에서 mouse / key 관련 callback 호출

현재 State 의 frameStarted 호출

현재 State 의 frameEnded 호출

Copyleft by 이대현

상태의 변경

```
void GameManager::changeState(GameState* state)
{
  if ( !states.empty() ) {
    states.back()->exit();
    states.pop_back();
  }
  states.push_back(state);
  states.back()->enter();
}
void GameManager::pushState(GameState* state)
{
  if ( !states.empty() ) {
    states.back()->pause();
  }
  states.push_back(state);
  states.back()->enter();
}
void GameManager::popState()
{
  if ( !states.empty() ) {
    states.back()->exit();
    states.pop_back();
  }
  if ( !states.empty() ) {
    states.back()->resume();
  }
}
```

현재 state의 exit() 함수 호출 – exit action 수행

현재 state 제거

다음 state를 top에

다음 state의 enter() 함수 호출 – entry action 수행

현재 state pause() 호출

다음 state를 top에 저장

다음 state의 enter() 호출 – entry action 수행

현재 state의 exit action

현재 state 제거

이전 state의 resume() 호출

Copyleft by 이대현

main.cpp의 분석

```
GameManager game;

try {

  game.init();
  game.changeState(TitleState::getInstance());
  game.go();

} catch( Ogre::Exception& e ) {
```

- 게임 관리자 초기화.
- 게임의 시작 상태를 TitleState로 지정.
- 현재 상태에서 게임 루프를 수행함.
- 내부적으로, 상태의 변화가 일어날 수 있음.

Copyleft by 이대현

TitleState.h 분석

```
class TitleState : public GameState
{
public:
  void enter();
  void exit();

  void pause();
  void resume();

  bool frameStarted(GameManager* game, const Ogre::FrameEvent& evt);
  bool frameEnded(GameManager* game, const Ogre::FrameEvent& evt);

  bool mouseMoved(GameManager* game, const OIS::MouseEvent &e)
  { return true; }

  bool mousePressed(GameManager* game, const OIS::MouseEvent &e, OIS::MouseButtonID id )
  { return true; }

  bool mouseReleased(GameManager* game, const OIS::MouseEvent &e, OIS::MouseButtonID id)
  { return true; }

  bool keyPressed(GameManager* game, const OIS::KeyEvent &e);
  bool keyReleased(GameManager* game, const OIS::KeyEvent &e) { return true; }

  static TitleState* getInstance() { return &mTitleState; }

private:

  static TitleState mTitleState;
  ... 후략 ...
```

mouse 처리 없으므로 true return

TitleState.cpp 에서 구현

Copyleft by 이대현

TitleState.cpp의 분석

```
TitleState TitleState::mTitleState;

void TitleState::enter(void)
{
  mContinue = true;
  mTitleOverlay = OverlayManager::getSingleton().getByName("Overlay/Title");
  mStartMsg = OverlayManager::getSingleton().getOverlayElement("StartMsg");
  mTitleOverlay->show();
}
void TitleState::exit(void)
{
  mTitleOverlay->hide();
}
bool TitleState::keyPressed(GameManager* game, const OIS::KeyEvent &e)
{
  switch(e.key)
  {
  case OIS::KC_SPACE:
    game->changeState(PlayState::getInstance());
    break;
  case OIS::KC_ESCAPE:
    mContinue = false;
    break;
  }
  return true;
}
```

class static 변수의 메모리 할당

Title Overlay 에서 "StartMsg" 요소 획득

TitleState를 빠져나가면 Title Overlay를 숨겨야 함

Copyleft by 이대현

PlayState.cpp의 분석

```
PlayState PlayState::mPlayState;

void PlayState::enter(void)
{
  ... 중략 ...
  mAnimationState = mCharacterEntity->getAnimationState("Run");
  mAnimationState->setLoop(true);
  mAnimationState->setEnabled(true);
}

void PlayState::exit(void)
{
  mSceneMgr->clearScene();
  mInformationOverlay->hide();
}
```

Empties the entire scene, inluding all SceneNodes, Entities, Lights, BillboardSets etc. Cameras are not deleted at this stage since they are still referenced by viewports, which are not destroyed during this process.

Copyleft by 이대현

학습 정리

■상태

- □ 오브젝트의 특정한 상황
  - ■ 이벤트에 의해서 상태가 변경
  - ■ Entry Action / Exit Action / Do Action을 수행

■State Pattern

- □ 스테이트 구현을 유연하게 할 수 있게 해주는 클래스 설계 방법
- □ 구현하고자 하는 상태를 추상 클래스 State로부터 상속받아, 구현함.
- □ 상태 관리자
  - ■ 현재 상태에 대한 Entry / Exit / Do Action을 수행함.
  - ■ 이벤트에 따라서 현재 상태를 변경함.

■State Pattern을 이용하여 Game Framework을 구현함으로써, 개발 효율성을 높일 수 있음.

Copyleft by 이대현