Strategy Design Pattern

Fonksiyonlar var, yeni yöntem ekleyeceksek bu pattern uygundur.

Farklı kütüphaneler kullanan farklı clientlar olsun. Birisi çizmek için OpenGL, biri DirectX vs. kullanıyor. Bir de serialize için farklı yöntemler eklendikçe işler karışır.

A diagram of a triangle and triangle

Description automatically generated

A diagram of a computer program

Description automatically generated

Bu böyle eklenen her yeni yöntem için uzayıp gidecek.

Shape’e nasıl draw ve nasıl serialize edeceğini söyle.

A diagram of a computer

Description automatically generated

Square ve Triangle stratejileri bilmeyecek.

Bu şekilde daha doğru. Böylece örneğin Circle eklersen artık OpenGLStrategy değişmez, OpenGLCircleStrategy eklersin, böylece Triangle ve Square yeniden compile olmaz.

OpenGLTriangleStrategy

OpenGLSquareStrategy

A diagram of strategy pattern

Description automatically generated

Command Design Pattern

Yapılacak işleri paketleyip, izole edip, uygun yerlere aktarmak.

A black background with yellow text and arrows

Description automatically generated

A diagram of a command pattern

Description automatically generated

std::function (functor, function pointer, lambda)

Game Loop:

switch (input) {

case MouseLeft:

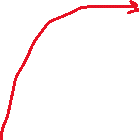
rotateViewLeft(); ViewPort::setViewAngle(…); … is angle

break;

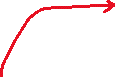
case MouseRight:

rotateViewRight();

break;



case MouseClick:

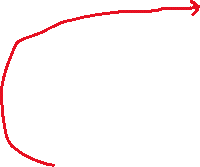


Animation::start  
Bullet::create  
Bullet::startMove

fireGun();

break;

};



Biri geldi, bu koda kaydetmeyi eklemek istiyorum onu ekle dedi. Hareketlerimizi store’a koyalım. Bu hareketler şöyle olabilir: (MoveLeft + parametreler + timestamp, MoveRight + parametreler + timestamp, …)

store()

A diagram of a company

Description automatically generated

Command 🡪 Çağrılış anındaki bütün bilgileri saklasınlar.

Invoked input (ex. MouseLeft) + Context (Mouse açısı, zaman, mouse hızı)

Her bilgiyi paketleyip command içerisine koyuyoruz ve sistemle komut ile haberleşiyoruz.

Game Loop:

switch (input) {

case MouseLeft:

c = createMLCommand();

c.execute();

history.push(c);

break;

case MouseRight:

c = createMRCommand();

c.execute();

history.push(c);

break;

case MouseClick:

c = createMCCommand();

c.execute();

history.push(c);

break;

};

Playback Loop:

…

for (auto c: history)

c.execute();