

Day2

1. Interacting window
2. Orientation
3. Rotate the Object

A better window for rendering

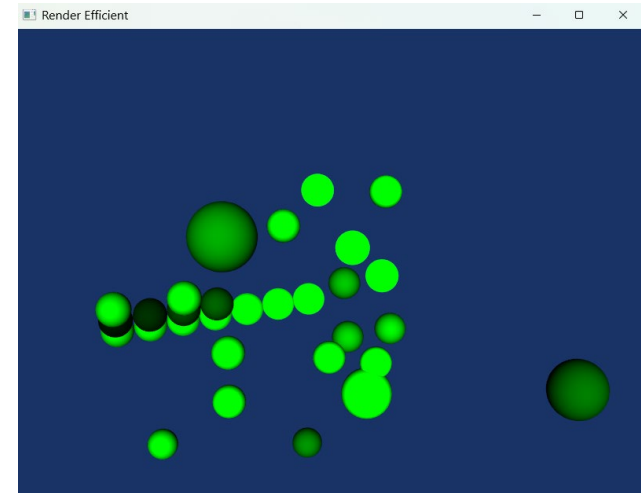
```
#include <vtkRenderWindowInteractor.h>
```

```
...
```

```
vtkNew<vtkRenderWindow> renWin;  
    renWin->AddRenderer(ren1);  
    renWin->SetSize(800, 600);  
    renWin->SetWindowName("Render Efficient");  
  
    vtkNew<vtkRenderWindowInteractor> iren;  
    iren->SetRenderWindow(renWin);  
    iren->Initialize();  
    iren->Start();  
}
```

Interact mode

- **Trackball Mode (t key):** Motion is strictly relative to mouse movement. The object stops moving when you stop dragging.
- **Joystick Mode (j key):** Continuous motion occurs based on mouse position relative to the screen center. The object spins continuously as long as you hold the mouse button.
- **Camera Mode (c key):** Mouse events manipulate the camera viewpoint around the focal point.
- **Actor/Object Mode (a or o key):** Mouse events physically translate, rotate, and scale the individual 3D objects or actors rather than moving the camera.



<https://vtk.org/doc/nightly/html/classvtkInteractorStyle.html>

- Keypress j / Keypress t: toggle between joystick (position sensitive) and trackball (motion sensitive) styles. In joystick style, motion occurs continuously as long as a mouse button is pressed. In trackball style, motion occurs when the mouse button is pressed and the mouse pointer moves.
- Keypress c / Keypress a: toggle between camera and actor modes. In camera mode, mouse events affect the camera position and focal point. In actor mode, mouse events affect the actor that is under the mouse pointer.
- Button 1: rotate the camera around its focal point (if camera mode) or rotate the actor around its origin (if actor mode). The rotation is in the direction defined from the center of the renderer's viewport towards the mouse position. In joystick mode, the magnitude of the rotation is determined by the distance the mouse is from the center of the render window.
- Button 2: pan the camera (if camera mode) or translate the actor (if actor mode). In joystick mode, the direction of pan or translation is from the center of the viewport towards the mouse position. In trackball mode, the direction of motion is the direction the mouse moves. (Note: with 2-button mice, pan is defined as <Shift>-Button 1.)
- Button 3: zoom the camera (if camera mode) or scale the actor (if actor mode). Zoom in/increase scale if the mouse position is in the top half of the viewport; zoom out/decrease scale if the mouse position is in the bottom half. In joystick mode, the amount of zoom is controlled by the distance of the mouse pointer from the horizontal centerline of the window.
- Keypress 3: toggle the render window into and out of stereo mode. By default, red-blue stereo pairs are created. Some systems support Crystal Eyes LCD stereo glasses; you have to invoke `SetStereoTypeToCrystalEyes()` on the rendering window.
- Keypress e: exit the application.
- Keypress f: fly to the picked point
- Keypress p: perform a pick operation. The render window interactor has an internal instance of `vtkCellPicker` that it uses to pick.
- Keypress r: reset the camera view along the current view direction. Centers the actors and moves the camera so that all actors are visible.
- Keypress s: modify the representation of all actors so that they are surfaces.
- Keypress u: invoke the user-defined function. Typically, this keypress will bring up an interactor that you can type commands in. Typing u calls `UserCallback()` on the `vtkRenderWindowInteractor`, which invokes a `vtkCommand::UserEvent`. In other words, to define a user-defined callback, just add an observer to the `vtkCommand::UserEvent` on the `vtkRenderWindowInteractor` object.
- Keypress w: modify the representation of all actors so that they are wireframe.

Axes

```
#include <vtkAxesActor.h>
```

```
void AddAxes(vtkRenderer* ren1) {  
    vtkSmartPointer<vtkAxesActor> axes = vtkSmartPointer<vtkAxesActor>::New();  
    std::string xyzLabels[3] = { "X", "Y", "Z" };  
    axes->SetShaftTypeToCylinder();// 設定軸的形狀為圓柱  
    axes->SetXAxisLabelText(xyzLabels[0].c_str());  
    axes->SetYAxisLabelText(xyzLabels[1].c_str());  
    axes->SetZAxisLabelText(xyzLabels[2].c_str());  
    axes->SetTotalLength(10.0, 10.0, 10.0);// 調整至合適比例  
    axes->SetCylinderRadius(0.5 * axes->GetCylinderRadius());  
    axes->SetConeRadius(1.025 * axes->GetConeRadius());  
    axes->SetSphereRadius(1.5 * axes->GetSphereRadius());  
  
    ren1->AddActor(axes);  
}
```

```

...
// Adding the orientation axes
AddAxes(ren1);

vtkNew<vtkRenderWindow> renWin;
renWin->AddRenderer(ren1);
renWin->SetSize(800, 600);
renWin->SetWindowName("Render Efficient");

vtkNew<vtkRenderWindowInteractor> iren;
iren->SetRenderWindow(renWin);

// =====
// 2. 設定左下角的 Orientation Marker Widget
// =====
vtkSmartPointer<vtkAxesActor> cornerAxes = CreateAxes();

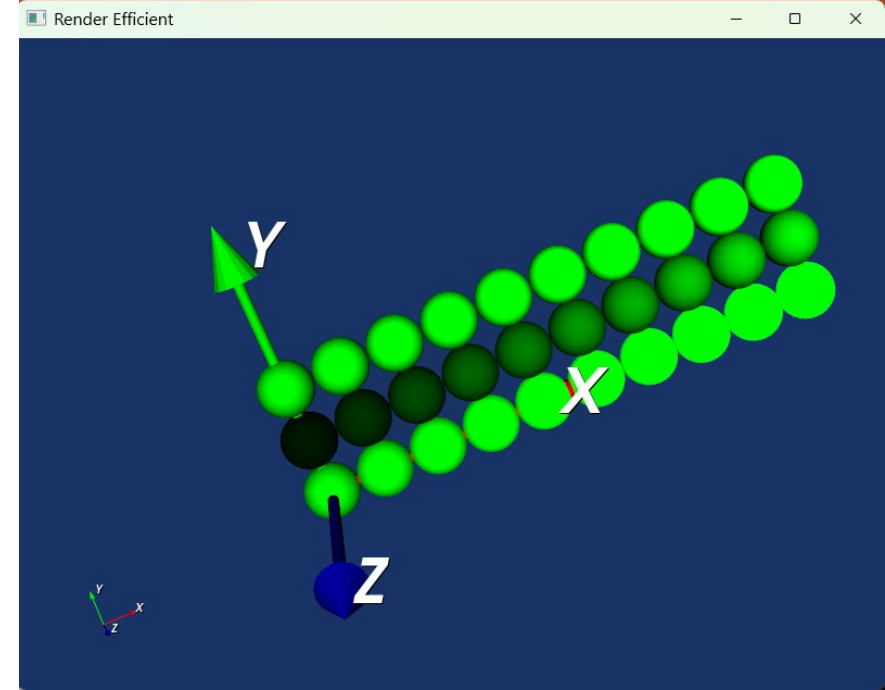
vtkNew<vtkOrientationMarkerWidget> widget;
widget->SetOrientationMarker(cornerAxes);
widget->SetInteractor(iren);
20% widget->SetViewport(0.0, 0.0, 0.2, 0.2); 區域
widget->SetEnabled(1);
widget->InteractiveOn();

```

```

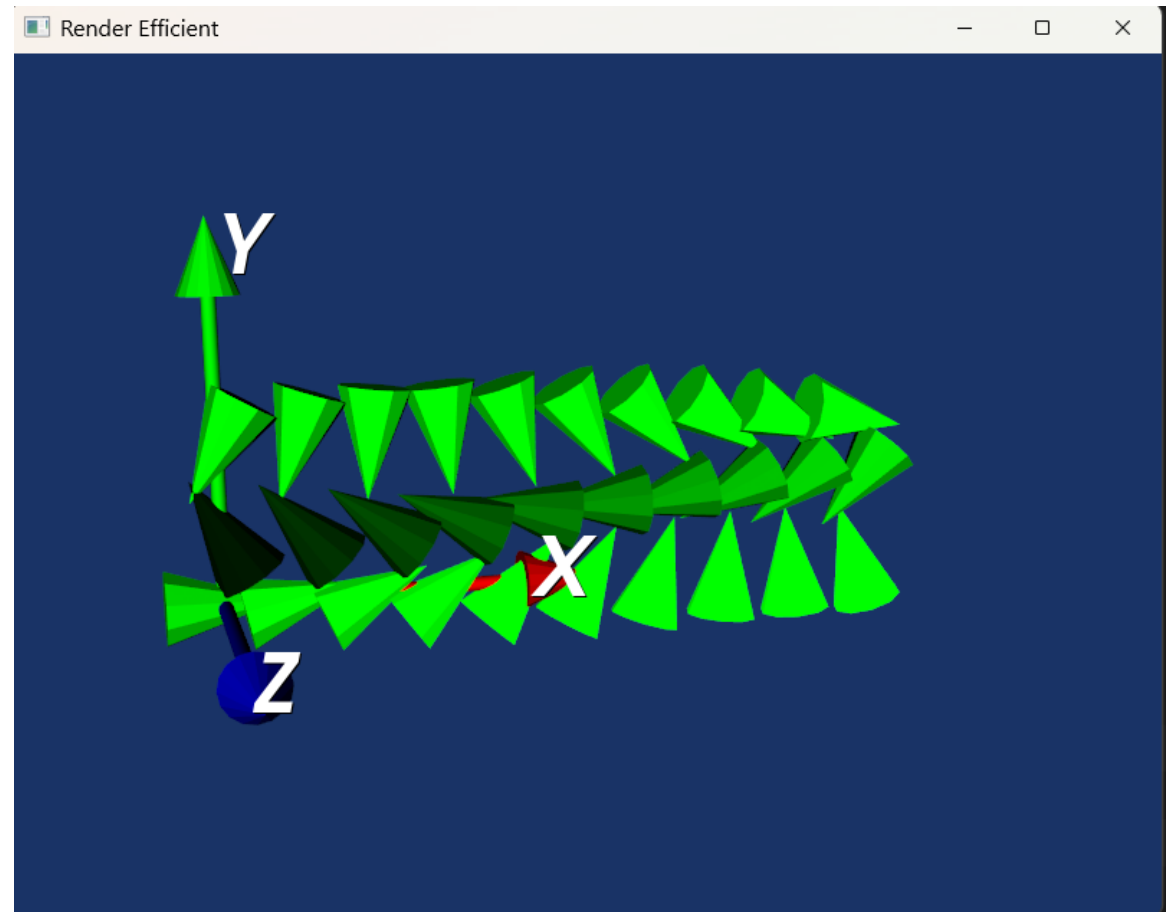
// 設定要在角落顯示的 Actor
// 綁定 Interactor
// 設定角落視埠位置 (xmin, ymin, xmax, ymax) -> 這裡設為左下角
// 啟用 Widget

```



Rotate the cone

```
...  
// Rotate the Objects  
for (int i = 0; i < actors.size(); ++i) {  
    double angle = 360.0 * i / actors.size();  
    actors[i]->RotateZ(angle);  
}  
...  
}
```



RotateTest

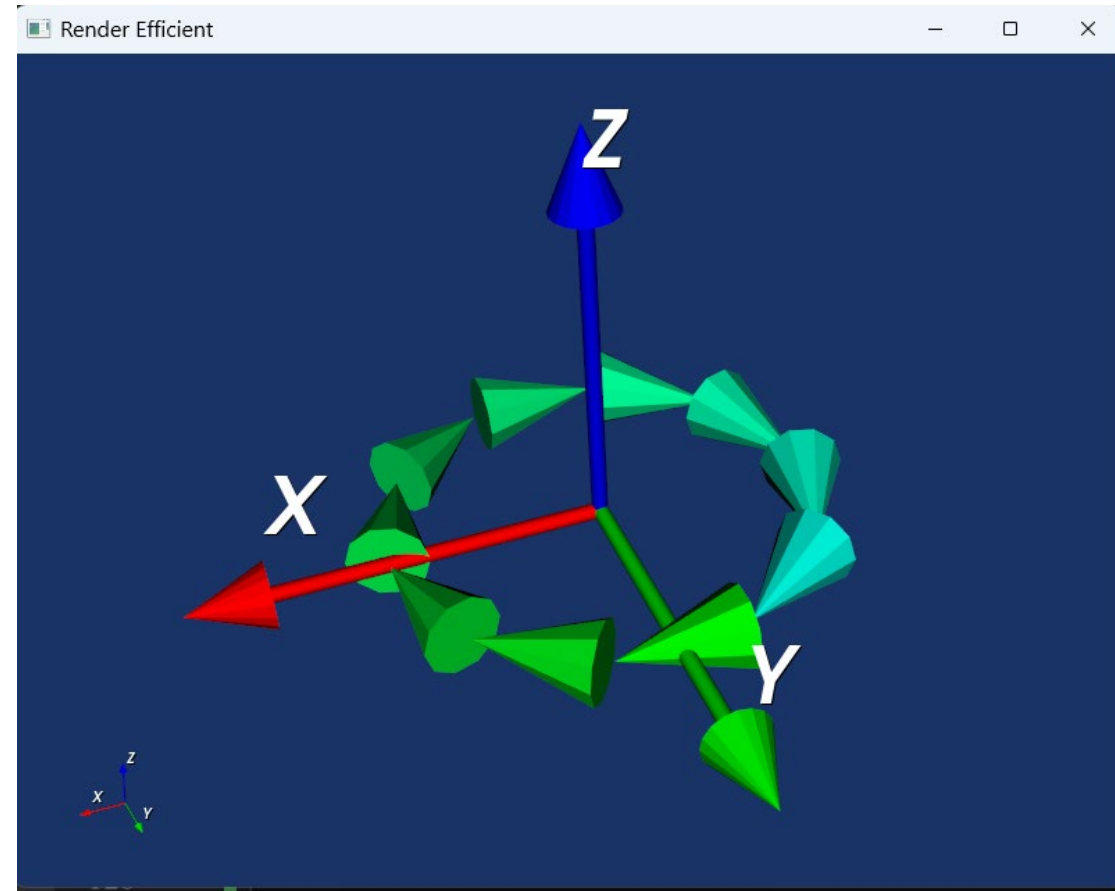
```
Actor->RotateX(degree);
```

```
Actor->RotateY(degree);
```

```
Actor->RotateZ(degree);
```

```
//
```

```
Actor->RotateWXYZ(degree, x, y, z);
```



RotateTest2: Let's try some changes

Tips: The height increases by 1 for each new object generated.

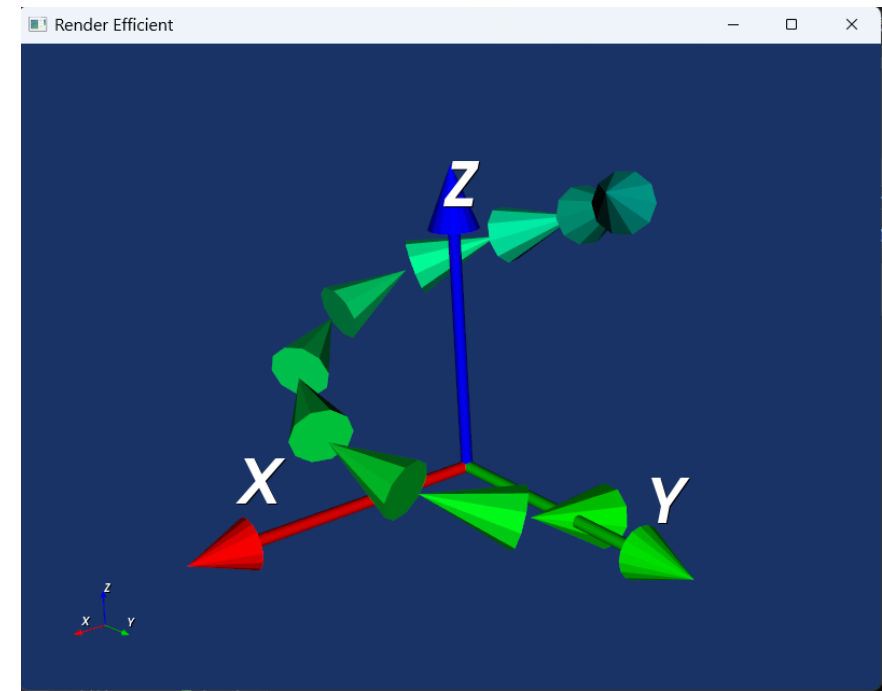
```
Double heightIncrease = 1;
```

Tips: Get the object orientation

```
double actorX = actor-&gtGetPosition()[0];  
double actorY = actor-&gtGetPosition()[1];  
double actorZ = actor-&gtGetPosition()[2];
```

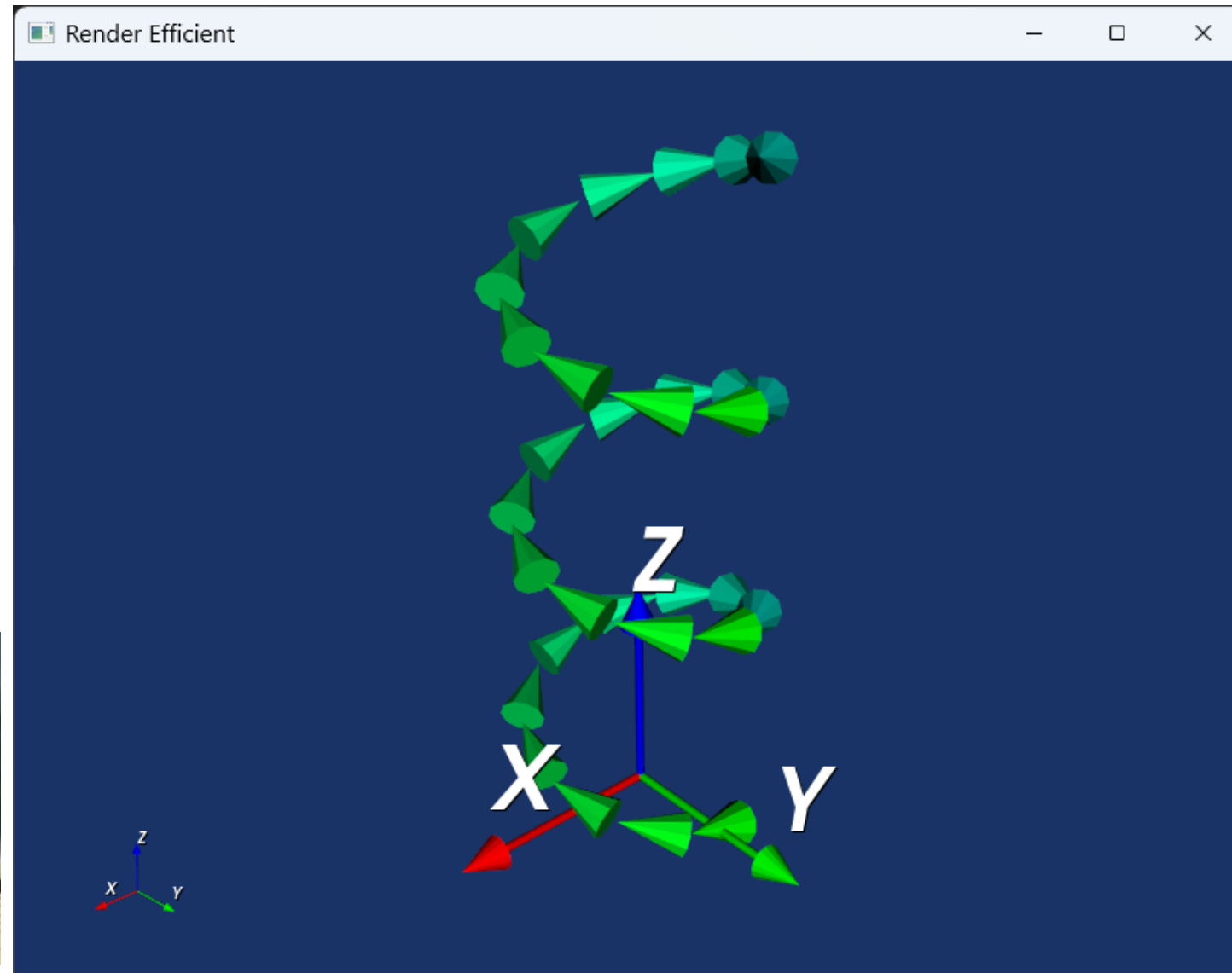
Tips: Use the function

```
actor->RotateWXYZ(degree, x, y, z);
```



RotateTest3: Tornado!

- Try it your self



HW: Try do some cool thing with the rotate function