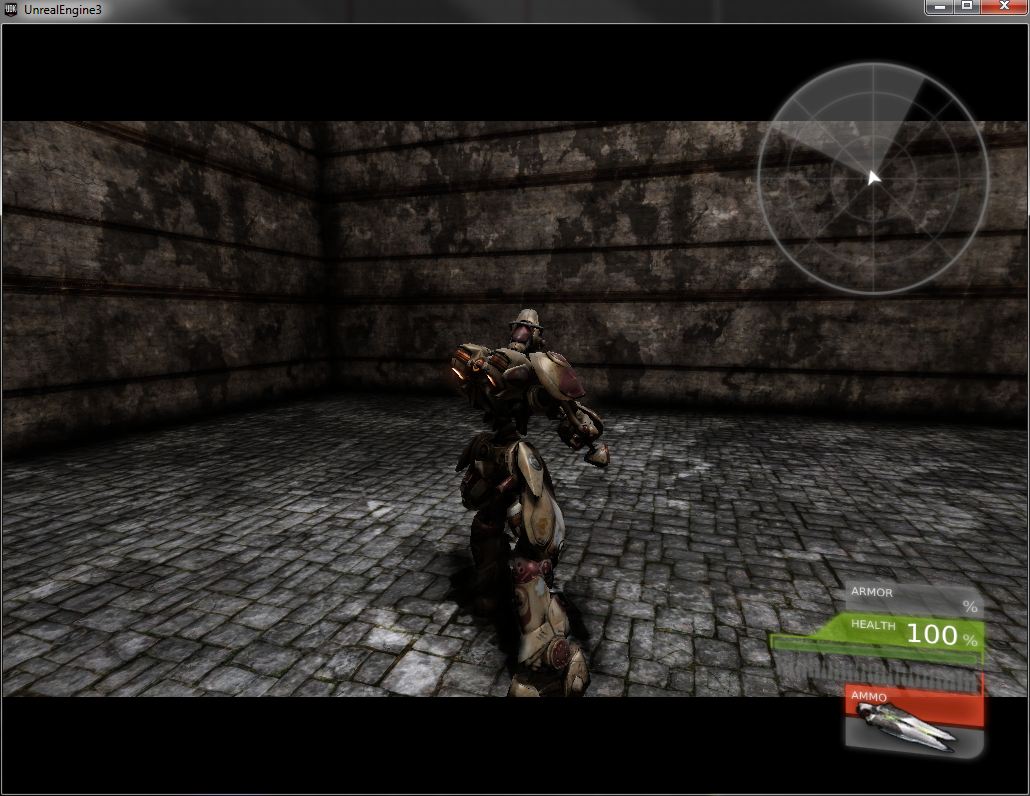
## Tương tác với đối tượng trong không gian.

Khi xây dựng chương trình trình diễn cảnh và các mô hình/đối tượng lấy từ vật thật trong không gian. Ngoài vấn đề di chuyển trong không gian, thay đổi góc nhìn và tương tác với các đối tượng trong cảnh, thì còn 1 vấn đề khác là chuyển từ xem cảnh trong không gian thành xem chi tiết đối tượng (lấy đối tượng làm trung tâm), xử lý các thao tác liên quan đến việc tương tác trực tiếp đến đối tượng. Trong chương trình của chúng em, các vấn đề gặp phải khi tương tác với đối tượng là xoay 1 hay toàn bộ đối tượng, di chuyển 1 hay toàn bộ đối tượng mà đảm bảo trong nhiều đối tượng cùng hiển thị, vẫn có 1 đối tượng được xác định là trung tâm.

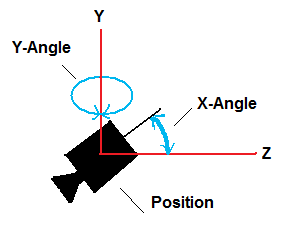
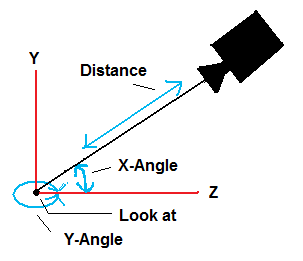


Sự khác nhau giữa góc nhìn thứ nhất và góc nhìn thứ ba.

Để giải quyết vấn đề này chúng em chọn 1 giải pháp là thay vì dùng cách thể hiện camera qua góc nhìn thứ nhất, đặt camera tại vị trí nhân vật, góc nhìn hướng ra cảnh xung quanh, thì sử dụng camera với góc nhìn người thứ 3. Khi đó camera quay quanh trọng tâm đối tượng và camera lúc nào cũng nhìn vào đối tượng này.

### Camera với góc nhìn người thứ 3 (third person camera)

Orbit camera là một khái niệm được nêu ra để áp dụng cho trường hợp sử dụng cho camera với góc nhìn người thứ 3. Orbit camera bao gồm các thành phần cơ bản như sau:

<http://www.ld4dstudio.nl/docs/manual/lightsCamsVars>

Góc nhìn thứ nhất và thứ ba, các thành phần cụ thể

* Target: Vị trí đối tượng trong không gian, camera dù ở hướng nào thì cũng có hướng nhìn vào vị trí này.
* Radius: Xác định khoảng cách trong không gian giữa Target và vị trí của camera. Khi giá trị này đợc xác định, ta nhận thấy, vị trí của camera lúc nào cũng nằm trên hình cầu có bán kính là Radius và có tâm là Target.
* Góc Alpha, Beta: Trong hệ trục tọa độ Orbit camera, không sử dụng thông tin x, y, z để xác định vị trí của camera trong không gian, mà sử dụng 2 góc để thể hiện. Cụ thể là 2 góc này là 2 góc quay quanh trục X, và quanh trục Y của đối tượng để xác định vị trí chính xác của Camera trên hình cầu.
* Camera Position: là kết quả tổng hợp tính toán từ vị trí Target, bán kính hìn hcầu Radius, góc Alpha và Beta.

var cosa = (float)Math.Cos(Alpha);

var sina = (float)Math.Sin(Alpha);

var cosb = (float)Math.Cos(Beta);

var sinb = (float)Math.Sin(Beta);

position = Target + new Vector3(Radius \* cosa \* sinb, Radius \* cosb, Radius \* sina \* sinb);

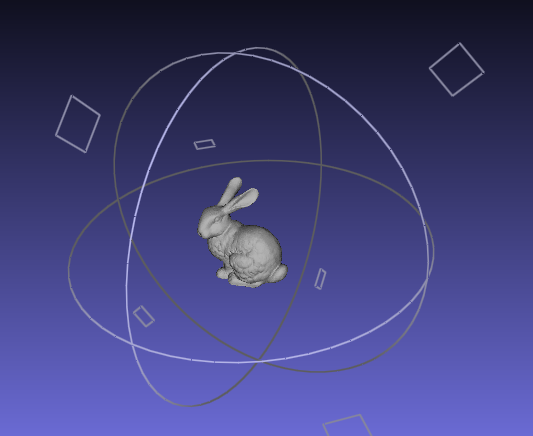
* View: ma trận chuyển đổi từ thế giới thực Oxyz sang hệ trục tọa độ của camera, thông tin được tính dựa trên Vector tính được từ vị trí Target và CameraPosition với Vector Up.

void UpdateViewMatrix()

{

view = Matrix.CreateLookAt(Position, Target, Vector3.Up);

}



meshlab sử dụng Orbit camera để xem model thỏ

### Xoay quanh đối tượng

Đối với hệ Orbit camera, vị trí tâm đối tượng được cố định, vị trí camera thay đổi và hướng vào tâm. Vị trí camera nằm trên hình cầu tâm là vị trí tâm đối tượng, nên camera quay quanh đối tượng thì cố định Radius, và Target, chỉ cóAlpha và Beta là thay đổi. Vấn đề chỉ còn là cập nhật góc Alpha và Beta phù hợp theo vị trí di chuyển của chuột trên màn hình..

Với mỗi lần di chuyển chuột hợp lệ (click chuột và move), ta xác định góc lệch cần cập nhật của Alpha và Beta:

Point currentPosition = e.GetPosition(element);

InertialAlpha += (float)(currentPosition.X - startPosition.X) \* AngularSpeed;

InertialBeta -= (float)(currentPosition.Y - startPosition.Y) \* AngularSpeed;

Trong đó:

startPosition, currentPosition là vị trí chuột cũ và mới trên màn hình.

AngularSpeed: góc lệch tương ứng với 1 pixel.

InertialAlpha, InertialBeta là góc lệch, cần cập nhật vào Alpha và Beta để tính lại vị trí mới Camera Position.

### Quay và tịnh tiến đối tượng.

Để thực hiện xoay đổi đối tượng quanh tâm, chúng em áp dụng cách tính góc quay thay đổi giống phần xx, tuy nhiên áp dụng thêm các bước để tính ma trận xoay 4x4 và áp dụng vào đối tượng phù hợp, các bước thực hiện như sau:

Bước 1: Xác định góc xoay delta Alpha và Beta, cộng vào Alpha và Beta, tính vị trí camera mới.

Bước 2: từ vector của Target với Camera Position cũ và vector của Target với Camera Position mới, ta xác định được ma trận quay R.

Microsoft.Xna.Framework.Vector3 NewCamPosition = \_3DPresentation.MathUtil.toNewCameraPosition(\_camera, dX, dY);

Microsoft.Xna.Framework.Vector3 OldPos = this.Camera.Target - this.Camera.Position;

Microsoft.Xna.Framework.Vector3 NewPos = this.Camera.Target - NewCamPosition;

Microsoft.Xna.Framework.Matrix lastMat = \_3DPresentation.MathUtil.GetTransformationMatrix(OldPos, NewPos);

Bước 3: Xác định tâm đối tượng muốn xoay (tâm đối tượng thường là trung bình cộng của vị trí giới hạn max min đối tượng trong không gian.

Bước 4: Xác định ma trận tịnh tiến T từ gốc tọa độ sang vị trí tâm đối tượng

Bước 5: áp dụng ma trận xoay này vào từng điểm của model muốn xoay bằng cách dời từng điểm về quanh tâm O, quay, rồi chuyển lại vị trí cũ:

Để di chuyển đối tượng, có 2 cách di chuyển:

Cách 1: Di chuyển theo hướng của camera và target, tức là camera xoay với mọi vị trí so với đối tượng thì, di chuyển tới, là di chuyển theo chiều vecor(0, 0, -1) trong hệ trục tọa độ camera.

Cách 2: Di chuyển theo hướng của thế giới thực, tức là với mọi góc nhìn và vị trí của camera, di chuyển tới là di chuyển theo chiều vector (0, 0, -1) trong hệ trục tọa độ thế giới thực.

Chúng em chọn cách 2 do cài đặt đơn giản, tiện cho người dùng. Do trong quá trình khảo sát, khi ghép frame lại với nhau, vửa xoay đối tượng vửa tịnh tiến thỉ dùng cách 2 dễ ghép hơn, đổi góc nhìn thì cùng 1 phím cùng 1 cách di chuyển, canh chuẩn hơn.

## Chọn đối tượng trong không gian

Trong quá trình đi chuyển trong cảnh, chúng em có nhu cầu tương tác với các đối tượng trong các cảnh đó (chọn đối tượng đế chuyển sang Page xem chi tiết đối tượng hoặc thêm bớt hiệu ứng tăng cường). Vấn đề là trong 1 cảnh bất kì, làm cách nào để khi click chuột vào đối tượng thì chọn được đối tượng, hoặc đúng đối tượng gần nhất.

Giải pháp chúng em đưa ra là sử dụng phương pháp kiểm tra giao nhau giữa 1 tia và đối tượng hình học đơn giản suy ra từ đối tượng - khung bao quanh đối tượng (bounding box), sau đó xác định đối tượng được chọn tốt nhất.

Minh hoạ cách chọn 1 model trong không gian.

Bước 1: Tính bounding box của từng đối tượng trong cảnh.

Bước 2: xác định tia R từ camera và đi qua vị trí trên màn hình (vị trí click chuột), dựa trên CameraPosition, View.

Ray ray = Babylon.Utilities.CreateRay((float)mouse.X, (float)mouse.Y, (float)drawingSurfaceSize.X, (float)drawingSurfaceSize.Y, Matrix.Identity, ActiveCamera.View, ActiveCamera.Projection);

Bước 3: Xác định khoảng cách từ vị trí camera đến vị trí cắt của tia R và các bounding box.

float? distance = ray.Intersects(mesh.BoundingInfo.BoundingBoxWorld);

Bước 4: Đối tượng có khoảng cách ngắn nhất là đối tượng được chọn.