

Java Racer

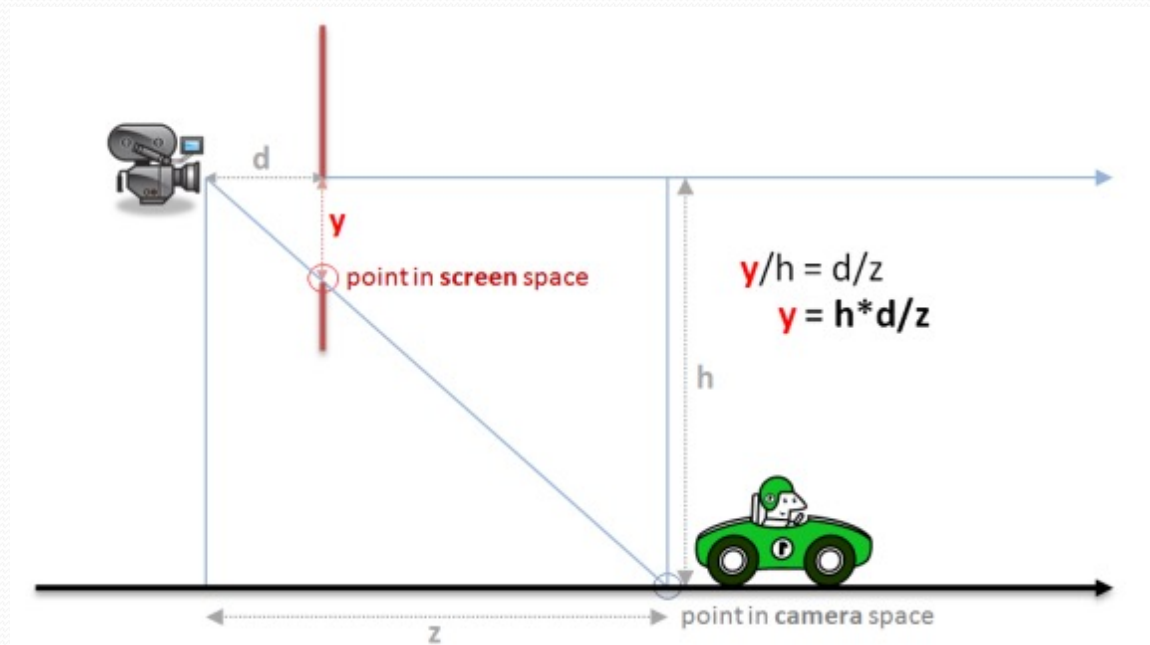
Versión 3 – carretera con colinas

Colinas

- En la versión 3 de Java Racer se le agregarán colinas a la carretera.



Proyección en perspectiva



Proyección en perspectiva

- Ecuaciones.

$$\begin{array}{l} \text{translate} \quad \begin{array}{l} x_{\text{camera}} = x_{\text{world}} - \text{cameraX} \\ y_{\text{camera}} = y_{\text{world}} - \text{cameraY} \\ z_{\text{camera}} = z_{\text{world}} - \text{cameraZ} \end{array} \end{array}$$

$$\begin{array}{l} \text{project} \quad \begin{array}{l} x_{\text{proj}} = x_{\text{camera}} * d/z_{\text{camera}} \\ y_{\text{proj}} = y_{\text{camera}} * d/z_{\text{camera}} \end{array} \end{array}$$

$$\begin{array}{l} \text{scale} \quad \begin{array}{l} x_{\text{screen}} = (w/2) + (w/2)*x_{\text{proj}} \\ y_{\text{screen}} = (h/2) - (h/2)*y_{\text{proj}} \end{array} \end{array}$$

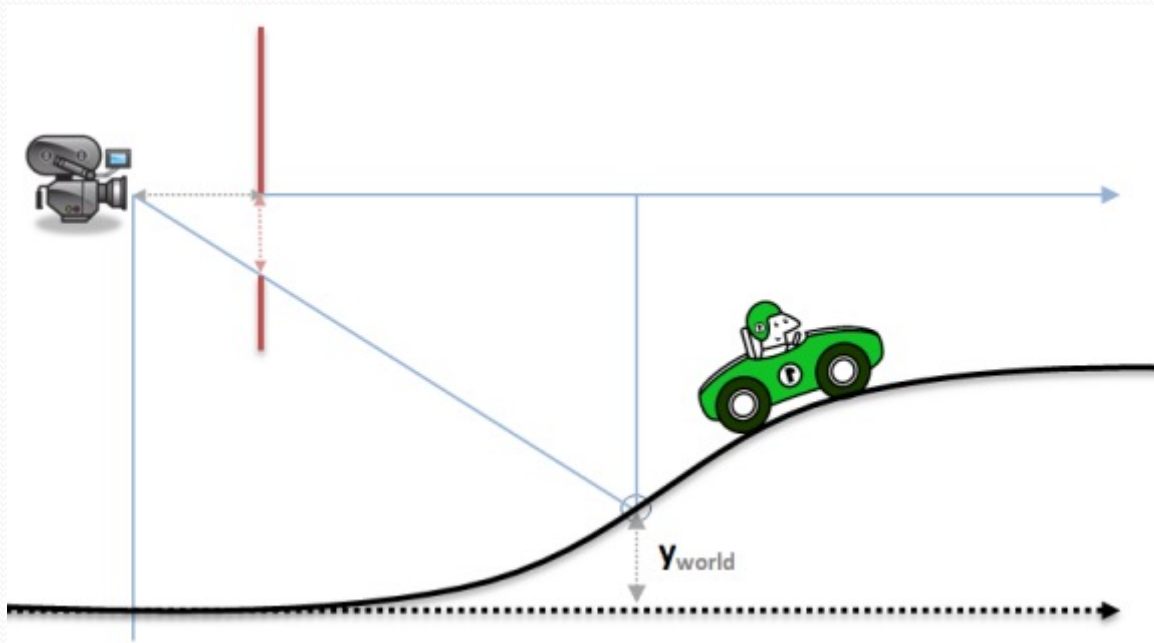
... where (w,h) = canvas **width** and **height**



Proyección en perspectiva

- En la versión 1 la carretera es recta.
- Solo se utiliza la coordenada z del mundo. Las coordenadas x y y del mundo son 0.
- En la versión 2 la carretera tiene curvas.
- La coordenada x del mundo no se utiliza porque las curvas son falsas (pseudo 3D).
- La coordenada y del mundo no se utiliza porque la carretera es plana.
- Para agregar colinas, la coordenada y del mundo debe ser no 0.
- A cada segmento de le debe agregar una coordenada y .

Proyección en perspectiva





Estructura del código

- Clases que cambian: `geometry.Road`, `render.Game`.
- Las demás clases no cambian.

Cambios en la geometría

- Se modifica el método addSegment.
- El valor de p2.world.y es un argumento.
- El valor de p1.world.y debe ser igual al valor de p2.world.y del segmento anterior.

Método addSegment modificado

```
private void addSegment(double curve, double y)
{
    int n = segments.size();
    segments.add(new Segment(
        n,
        new PointProjection(new Point3D(0, lastY(), n * segmentLength), new
Point3D(), new Point4I()),
        new PointProjection(new Point3D(0, y, (n + 1) * segmentLength), new
Point3D(), new Point4I()),
        curve,
        (n / rumbleLength) % 2 == 1 ? Colors.DARK : Colors.LIGHT));
}
```

Método lastY

```
private double lastY()
{
    return segments.isEmpty() ? 0 : segments.get(segments.size() -
1).p2.world.y;
}
```



Clase Road modificada

- Se agrega otro objeto enum para representar las colinas.

Clase Road modificada

```
public enum HILL {  
    NONE(0), LOW(20), MEDIUM(40), HIGH(60);  
    private final int hill;  
    HILL(int hill) {  
        this.hill = hill;  
    }  
    public int getValue() {  
        return hill;  
    }  
}
```



Método addRoad modificado

- Se modifica el método addRoad para aceptar un parámetro y.
- Este argumento se usa en las transiciones para entrar y salir de las colinas.

Método addRoad modificado

```
private void addRoad(double enter, double hold, double leave, double curve,  
double y)  
{  
    double startY = lastY();  
    double endY = startY + y * segmentLength;  
    double total = enter + hold + leave;  
    for (int n = 0; n < enter; n++) {  
        addSegment(Util.easeIn(0, curve, n / enter), Util.easeInOut(startY,  
endY, n / total));  
    }
```

Método addRoad modificado

```
for (int n = 0; n < hold; n++) {  
    addSegment(curve, Util.easeInOut(startY, endY, (enter + n) / total));  
}  
for (int n = 0; n < leave; n++) {  
    addSegment(Util.easeInOut(curve, 0, n / leave), Util.easeInOut(startY,  
endY, (enter + hold + n) / total));  
}  
}
```

Agregando colinas

```
private void addHill(int num, int height)
{
    addRoad(num, num, num, 0, height);
}
```

Agregando colinas

- Se pueden agregar varias colinas a la vez.

```
private void addLowRollingHills(int num, int height)
{
    addRoad(num, num, num, 0, height / 2.0);
    addRoad(num, num, num, 0, -height);
    addRoad(num, num, num, Road.CURVE.EASY.getValue(), height);
    addRoad(num, num, num, 0, 0);
    addRoad(num, num, num, -Road.CURVE.EASY.getValue(), height / 2.0);
    addRoad(num, num, num, 0, 0);
}
```



Cambios en el método update

- En este juego no se intenta simular la realidad.
- Las colinas no afectan al jugador y no se necesitan cambios en update.



Cambios en el método render

- Desde la versión 1 las ecuaciones de proyección están preparadas para valor de y distintos de cero.
- Se cambia el renderizado del fondo para permitir parallax scrolling vertical.
- Las capas del fondo se mueven verticalmente conforme el jugador entra y sale de una colina.
- En la proyección ahora hay que tomar en cuenta la coordenada y del jugador.
- También al renderizar el automóvil.

Cambios en el método render

```
private void render(Graphics2D g2)
{
    ...
    Segment playerSegment = findSegment(position + playerZ);
    double playerPercent = Util.percentRemaining(position + playerZ,
segmentLength);
    double playerY = Util.interpolate(playerSegment.p1.world.y,
playerSegment.p2.world.y, playerPercent);
    ...
}
```

Cambios en el método render

Render.background(g2, background, width, height, Background.SKY, skyOffset, **resolution * skySpeed * playerY**);

Render.background(g2, background, width, height, Background.HILLS, hillOffset, **resolution * hillSpeed * playerY**);

Render.background(g2, background, width, height, Background.TREES, treeOffset, **resolution * treeSpeed * playerY**);

...

Cambios en el método render

```
for (int n = 0; n < drawDistance; n++) {
```

```
...
```

```
    Util.project(segment.p1, (playerX * roadWidth) - x, playerY +  
cameraHeight, position - (segment.looped ? trackLength : 0), cameraDepth,  
width, height, roadWidth);
```

```
    Util.project(segment.p2, (playerX * roadWidth) - x - dx, playerY +  
cameraHeight, position - (segment.looped ? trackLength : 0), cameraDepth,  
width, height, roadWidth);
```

```
...
```

```
}
```

Cambios en el método render

```
Render.player(g2, width, height, resolution, roadWidth, sprites,  
    speed / maxSpeed,  
    cameraDepth / playerZ,  
    width / 2.0,
```

```
    (height / 2.0) - (cameraDepth / playerZ *  
Util.interpolate(playerSegment.p1.camera.y, playerSegment.p2.camera.y,  
playerPercent) * height / 2),  
    speed * (keyLeft ? -1 : keyRight ? 1 : 0),  
    playerSegment.p2.world.y - playerSegment.p1.world.y);
```

Resultado

- La versión 3 agregó colinas reales a las curvas falsas.

