Project Report AIVG
DrunkStride

Anthony Baiamonte; m. 28792A

18 January 2024

1 Introduction

The goal of the project was to create an agent roaming on a platform while never moving along a straight line.
The project specifications were as follows:

- The platform can be of any size: square or rectangular;
- The Agent will change trajectory every random interval of (0,10] seconds;
- The Agent is moving at a constant speed of $1\frac{m}{s}$;
- Every time interval the Agent will travel over e circumference leading first to right then to the left, and so on;
- The Agent will select a random circumference leading right or left with a radius between 0 (excluded) and the maximum radius that isn't making the Agent fall off the platform;
- The selection of the radius is independent from the time of the next trajectory change;
- The agent never stop moving;
- The system must work independently on the platform shape or size.

2 Design of the AI

To make sure that the Agent is never moving on a straight line, we make the Agent move on a circumference. Initially the idea was to generate a circumference in front of the Agent like we can see in the figure 1. But with this setting there were some problem:

1. determine if the circle was inside the bound of the platform;
2. make the Agent do a smooth turn when there was a change of the rotation direction, how we can see in the figure 3;

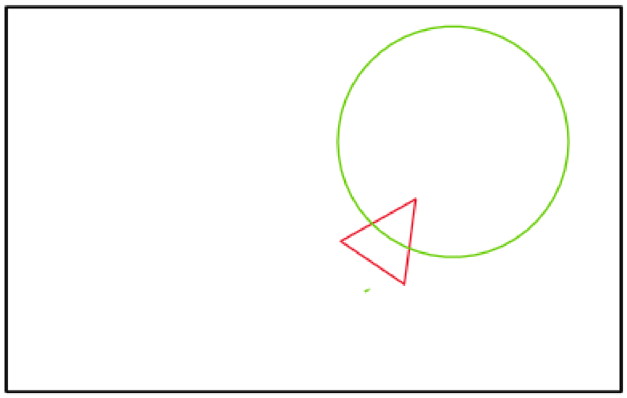


Figure 1: Representation of the idea.

Regarding the first problem mentioned. The first method use to determine if the circle was inside the bounds of the platform was to control if the 3 extreme point (forward, left and right) where inside the platform. But with this method there were some cases where even if all these 3 point were inside the platform, some other point of the circle where outside, like we can see in the figure 2.

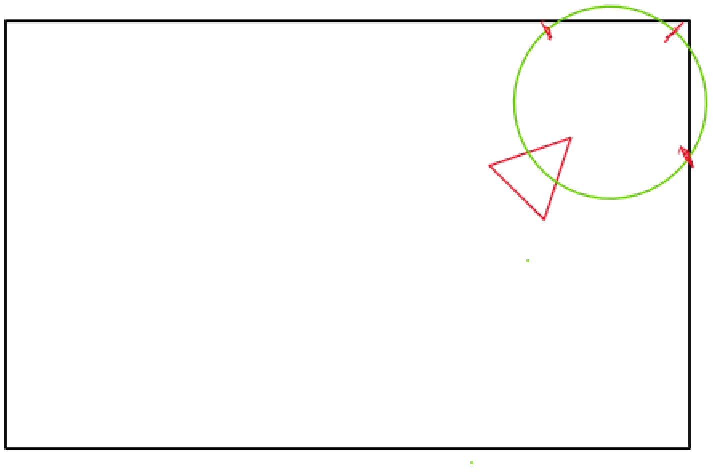


Figure 2: Demonstration of the case where the 3 extreme are inside the platform but other point on the circle not.

This problem was solved by simply checking more that 3 point on the circle, but a good number of them, generating a series of angle values on the circumference and checking whether that point was inside the platform or not.

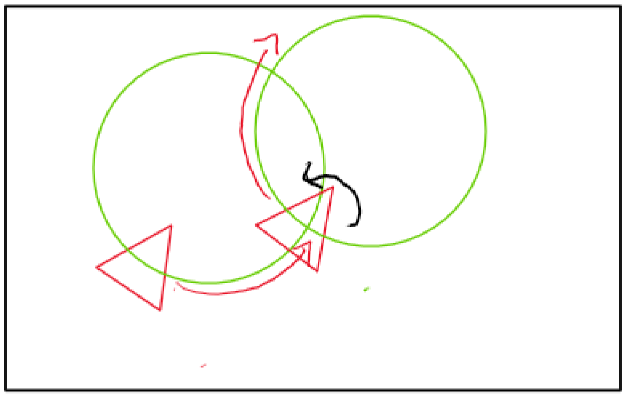


Figure 3: smooth rotation towards the direction of movement.

Regarding the second problem mentioned. The design specification required rotating first right and then left on a circumference, so clockwise and counterclockwise. In doing these, some changes were necessary to make turn the the Agent towards the tangent of the circle, and make it look toward the movement direction. In doing these rotation there where some problem of smoothness of the movement so i decided to change the approach (fig. 4).

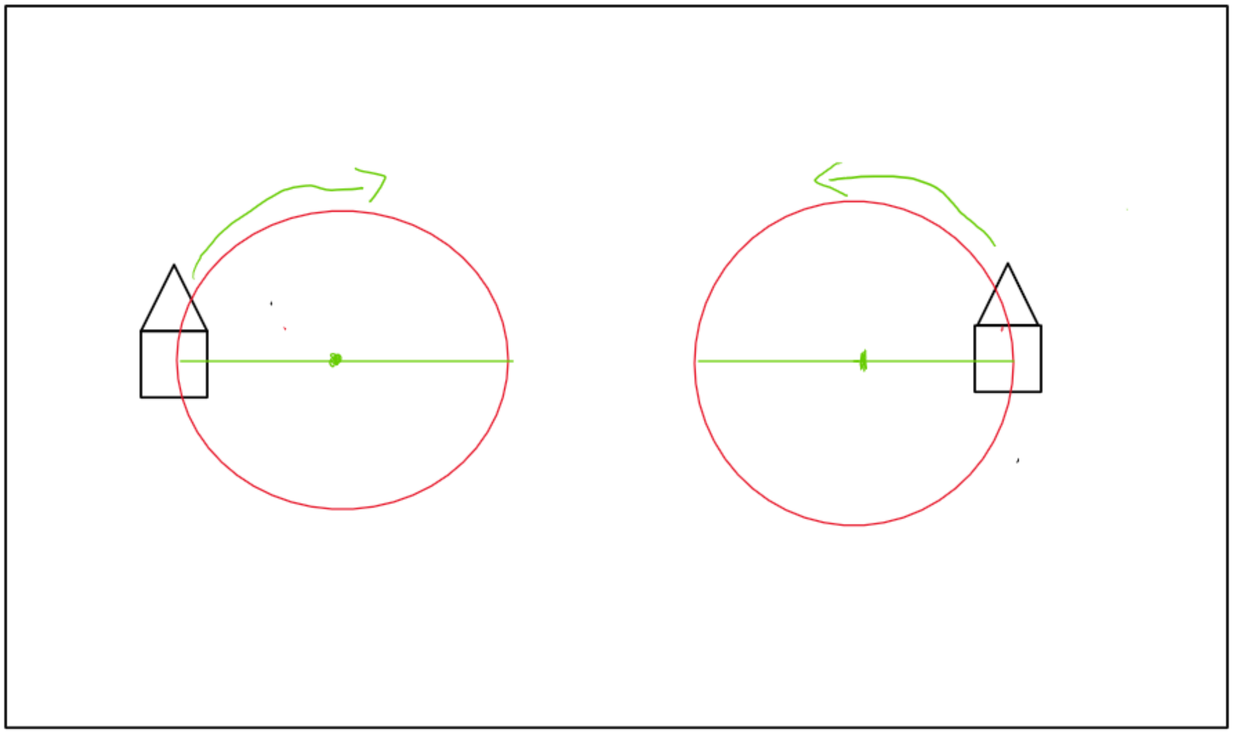


Figure 4: New configuration of the circumference.

In these new configuration the circumference where generated at right or left of the Agent, according to the rotation orientation, clockwise or counterclockwise. If the Agent were to turn clockwise then the circle was generated at the right of the Agent, and if the agent was to turn counterclockwise then the circle was generated to the left of the agent. Using these method the change of trajectory are a lot more Smooth (fig. 5).

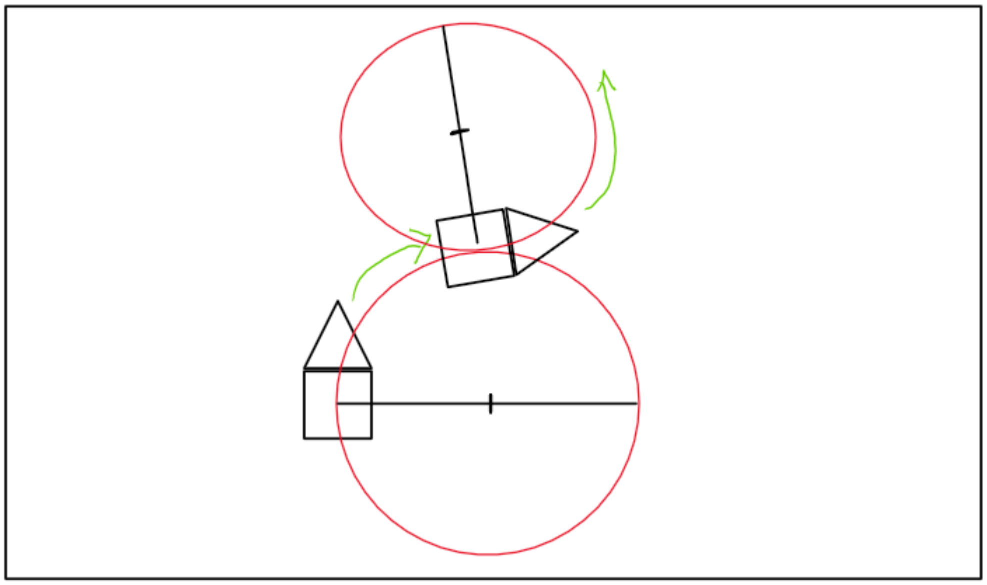


Figure 5: Turning smoothly.

Now let's see the method used to determine the Max radius that is being used as max value to generate random the radius of the circumference on which the Agent in moving. Here i decide to use two different approach.

- One approach consist in calculate incrementally the radius of the circle and controlling if the generated circle is inside the platform (fig. 6);
- The other is to generate immediately the maximum possible radius control if the circle is correct, ad if necessary decrement the value until the value is right (fig. 6).

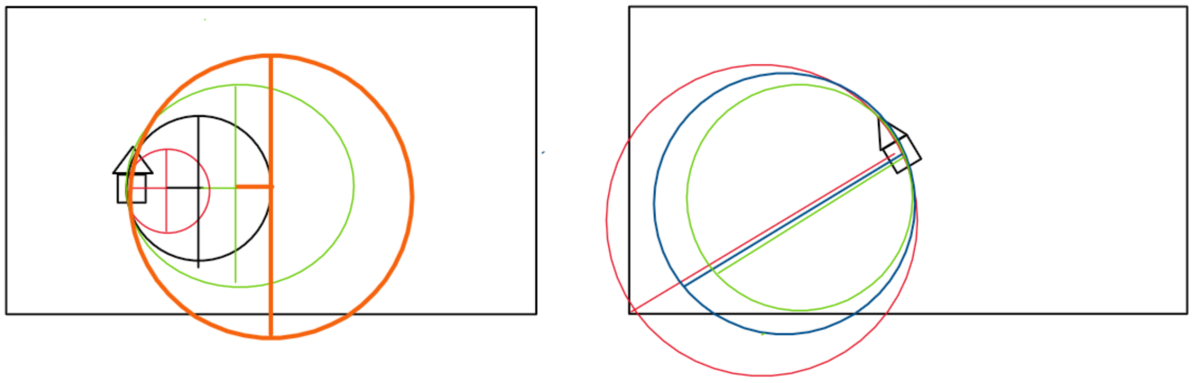


Figure 6: Caption

Each of the two method has it's pros and cons, which we will now list.

Pros.

- The pros of the first method is that in a very small platform, or in platform that is very similar to an hallway, this method in faster than the second, keeping in mind the orientation of the Agent. As we can see in the figure 7, with the first method we obtain relatively quickly the green circle;
- The pros of the second method is that with very large platform we obtain the approximate maximum radius much faster that the first method, assuming we are in a general position on the plan;

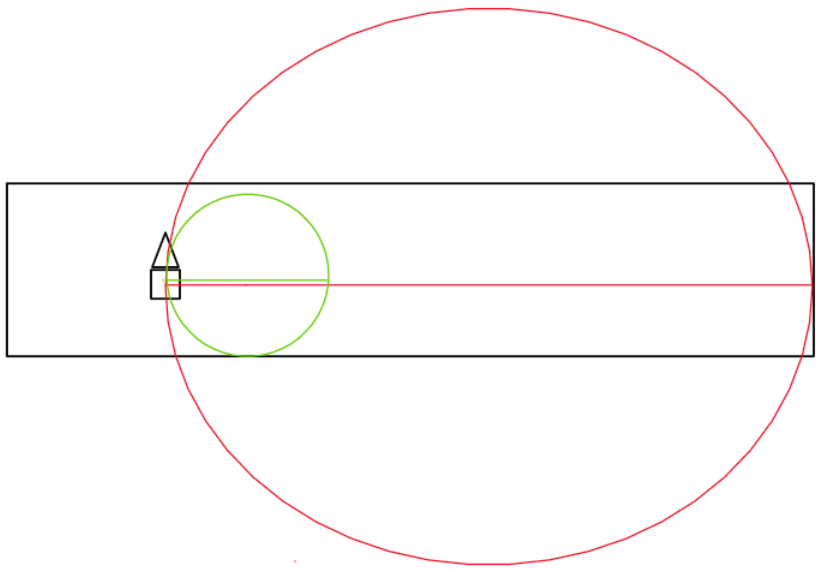


Figure 7:

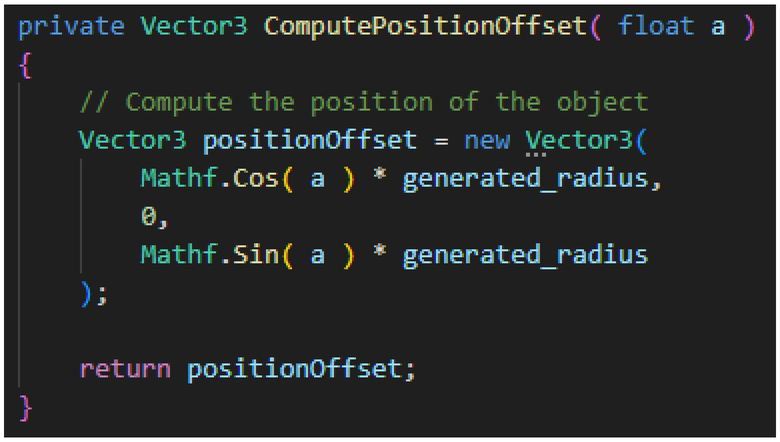
Cons.

- The cons of the first method is that in very large map we take more time to calculate the max radius to generate the circle;
- The cons of the second method is the one that we can see in the figure 7. Based of the orientation and the verse of the rotation, this method can return a max radius that doesn't make a good representation of the situation. So we will pass some time to adjust the value of the obtained radius.

3 Implementation of the AI

To make move the Agent we chosen the **kinematic** way. We operate only on the transform of the Agent without using the method of a RigidBody, and there is no particular reason for these choice, except for having a fixed speed. So, using one of the two method specified in the previous section, we calculate the maximum radius of the current time interval, and then we make move the Agent on the generated circumference.
To calculate the correct position on the circle, first we calculate the center of the circle. Considering a

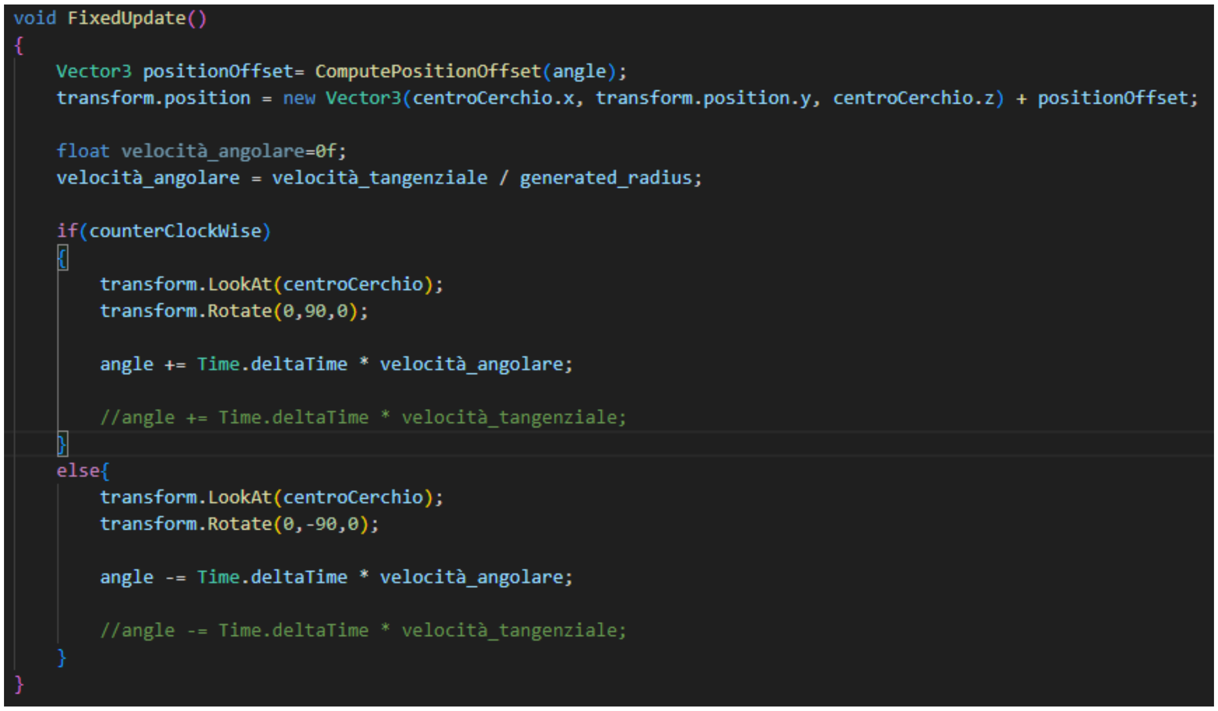
circle centered in (0,0) with the radius we calculated, with some trigonometric we can calculate the position on the circumference, and the we add this point to effective center of the circle we calculate before. As result we have now the current position on the generated circle. Going forward we keep incrementing (or decrementing, it depends on the sense of the rotation) the value of the angle, being careful to use the right values for the angle every time there is a change in rotation, passing from clockwise to counterclockwise and vice versa. The effective position of the Agent on the circle is calculated in the method fixedUpdate, and



```
private Vector3 ComputePositionOffset( float a )
{
    // Compute the position of the object
    Vector3 positionOffset = new Vector3(
        Mathf.Cos( a ) * generated_radius,
        0,
        Mathf.Sin( a ) * generated_radius
    );

    return positionOffset;
}
```

Figure 8:



```
void FixedUpdate()
{
    Vector3 positionOffset= ComputePositionOffset(angle);
    transform.position = new Vector3(centroCerchio.x, transform.position.y, centroCerchio.z) + positionOffset;

    float velocità_angolare=0f;
    velocità_angolare = velocità_tangenziale / generated_radius;

    if(counterClockWise)
    {
        transform.LookAt(centroCerchio);
        transform.Rotate(0,90,0);

        angle += Time.deltaTime * velocità_angolare;

        //angle += Time.deltaTime * velocità_tangenziale;
    }
    else{
        transform.LookAt(centroCerchio);
        transform.Rotate(0,-90,0);

        angle -= Time.deltaTime * velocità_angolare;

        //angle -= Time.deltaTime * velocità_tangenziale;
    }
}
```

Figure 9:

we can see in the figure 9, we use the **LookAt** method and **Rotate** to make sure that the Agent is always facing in the direction of the movement, that is along the tangent of the circle, always taking in to account the verse of the rotation.

To control if the generate circle is inside the boundaries of the platform we use a mixture of different method. First we generate a **Bounds** object, and thanks to this object we can easily control if some point is inside the bounds or not. Then we take some sampling point on the circle generate and we control if this point are inside the boundaries. This control is repeated on the maximum radius generate until we are sure that every circle with radius between the minimum and the maximum are entirely inside the boundaries.

We can say something about the two method use to generate the presumed maximum radius.

1. The first method consist only in incrementing the value of the maximum Radius until we find a circle that is outside of the platform, nothing more. For every radius, we generate the circle and check it.

2. The second method uses directly two Unity tool, the **BoxCast** and the the **Physics.RayCast method** (fig. 11). Doing a Ray cast, for the correct distance, and a suitable direction for the verse of the rotation, we look for an hit of the ray. The position of the hit is the is the presumed maximum radius (we can see the structure of the BoxCast in the figure 10).

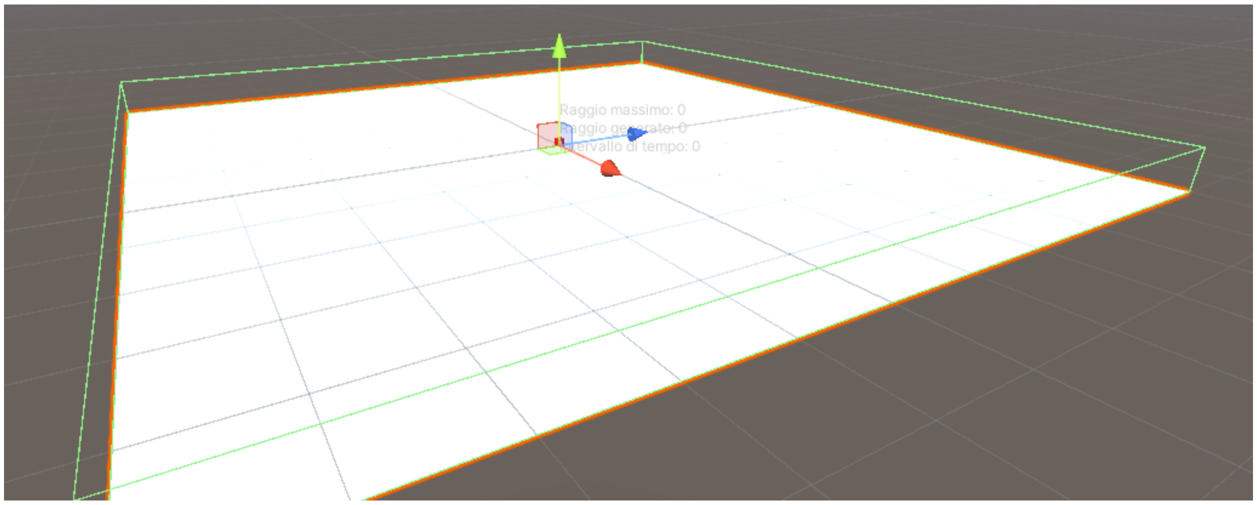
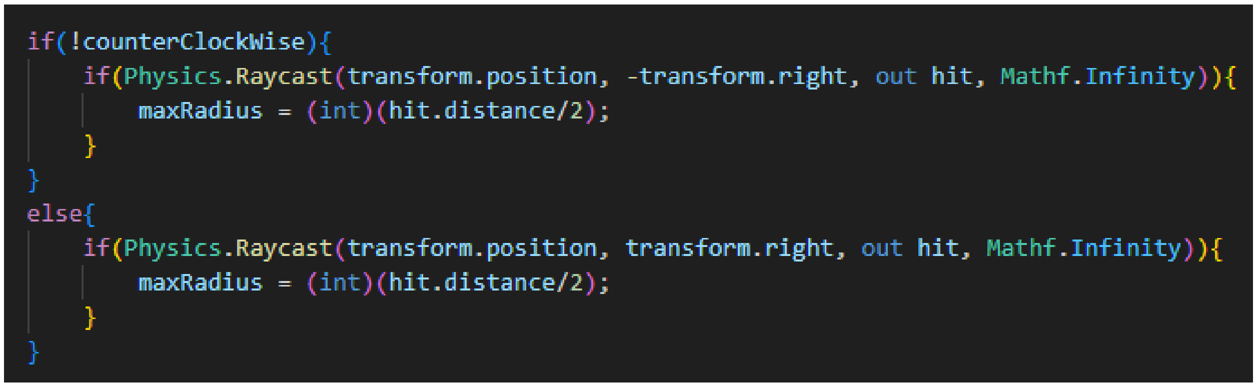


Figure 10:



```
if(!counterClockWise){
    if(Physics.Raycast(transform.position, -transform.right, out hit, Mathf.Infinity)){
        maxRadius = (int)(hit.distance/2);
    }
}
else{
    if(Physics.Raycast(transform.position, transform.right, out hit, Mathf.Infinity)){
        maxRadius = (int)(hit.distance/2);
    }
}
```

Figure 11:

Even if we try to generate the maximum radius that does not cause the Agent to fall from the platform, the Agent still has its own size, and the situation where the Agent is too close to the border and then fall can always happen. To make sure that the Agent never stop moving on a circumference we generate a radius that is never less than 1. Taking in consideration this minimum radius of 1, if at some moment there isn't the possibility to generate a new radius at least of 1, then the Agent continue to move on the same circumference until the time interval expire. This choice of the minimum radius was simply dictated by an aesthetic question and taking into account the dimension of the agent, that in this case i a cube. The requirement was that the agent never move in a straight line, but with too small a radius, it would appear as if it were spinning around itself. For this reason them movement implemented doesn't work too well with a platform that has a local scale inferior to $(0.5, 1, 0.5)$.
There is no function to dynamically change the size of the platform, since it is not required by the specifications. Despite this it could be easily added, by recalculating the size of the plane periodically, thus adapting the controls of the maximum radius generated. Furthermore, in this case one should be careful of the situation in which changing the size could lead to the agent falling into the void.

4 Examples

In the last two image we can see the result of the execution. The figure 12 is obtain we the first method of generation of the radius. The figure 13 it is instead obtained with the second method, and the black line that is present in the image represent the Ray cast done to see which is the maximum radius (you can see in the figure the the maximum is 23 and the generated is 4).

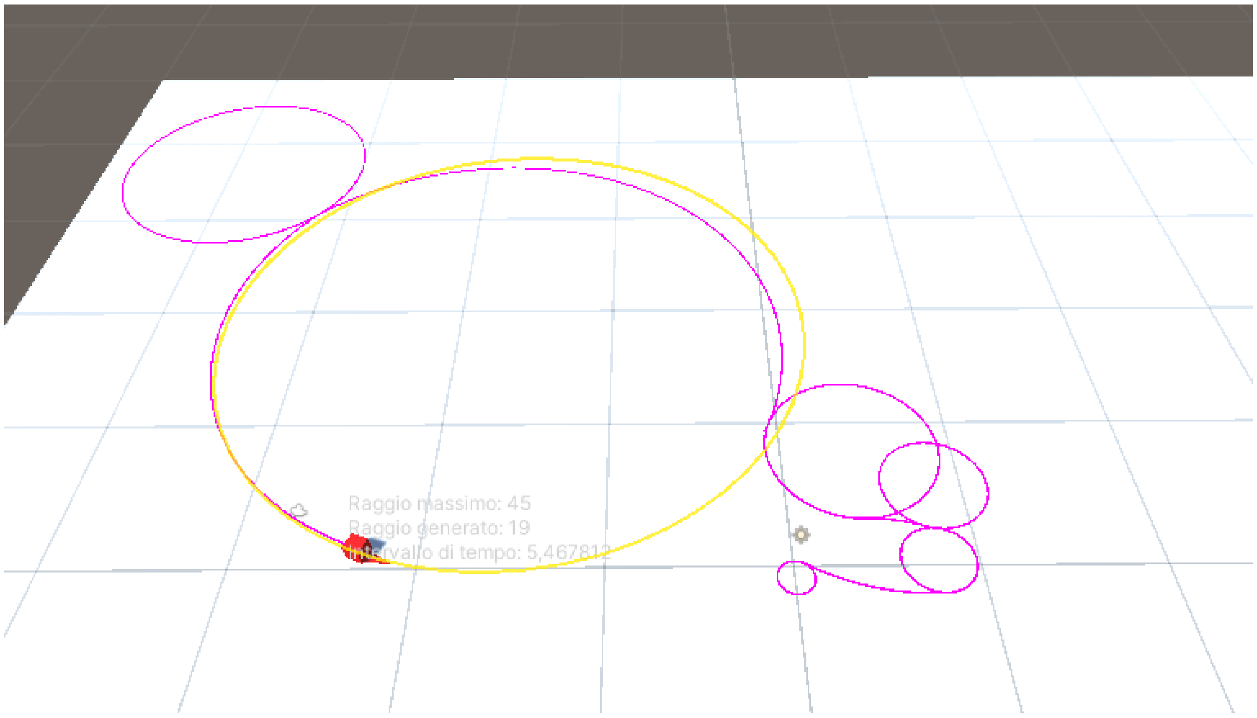

Raggio massimo: 45
Raggio generato: 19
Intervallo di tempo: 5,467812

Figure 12: Little example of the execution with the first method.

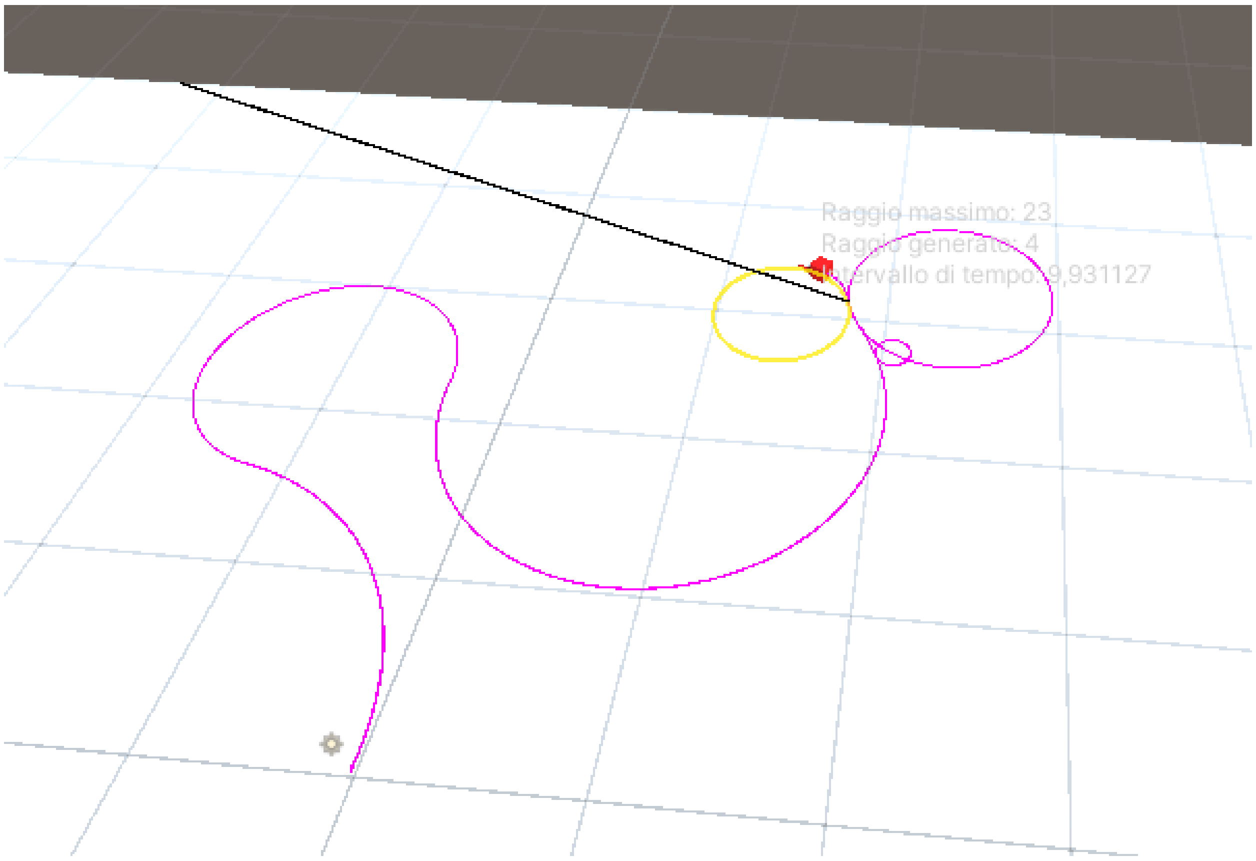


Figure 13: Little example of the execution with the second method.