AUTHOR	**Cardboard Buddies**
CONTACT	**michael.soler.beatty@gmail.com**
Unity Ver.	**2018.3.f1**

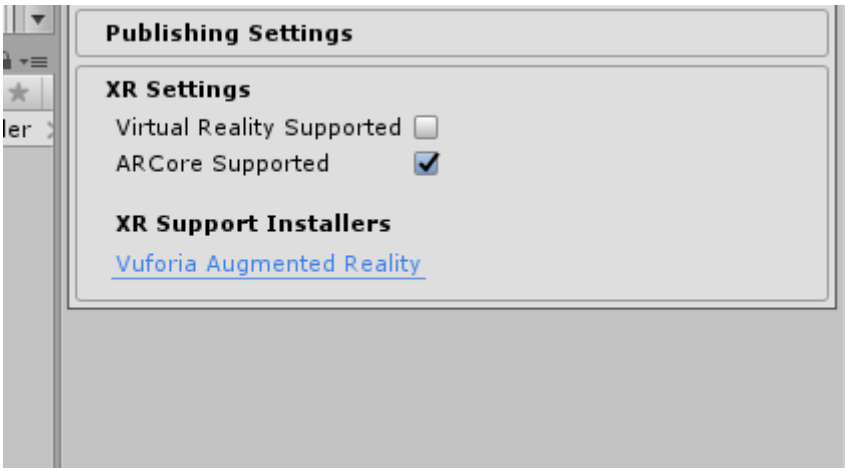
Index

1.Dependencies

This package needs one main package that is called ARcore:

https://github.com/google-ar/arcore-unity-sdk/releases

Remember to enable ARcore in the XR settings.



2.Description of the package.

This package allows the player to generate a bowling environment. The aim of the game is to score by knocking down pins. TO achieve this, the player has to throw a ball in different rolls and frames. It works with ARCore and touching events on Android. A smooth filter is used for the computation of speed of the rigidbody attached to the ball, which is the basis for its movement once the player releases the screen.

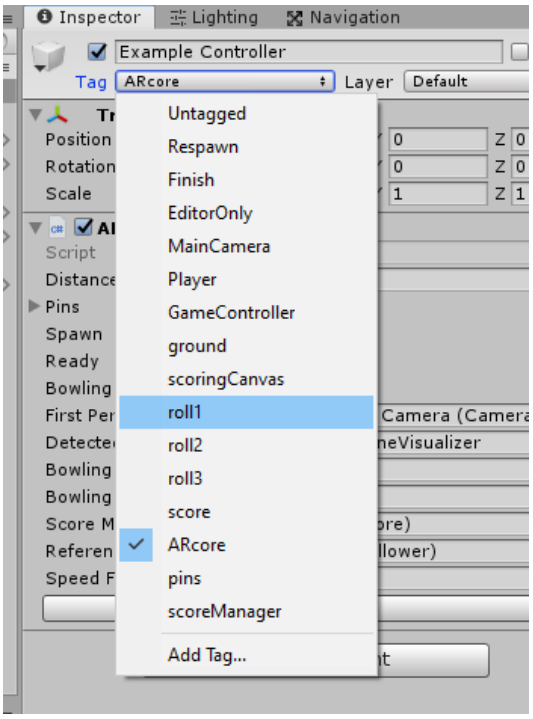
The package's work-flow can be summarized here:

1. Import ARCore.
2. Install ARCore in your Android Device.
3. Import our AR bowling package.
4. Export to Android with XR setting set to true.
5. Tap on the detected plane in AR to create the pins and the bowling environment.
6. Drag on the screen and release it to throw the balls in each roll.
7. Wait for the score to compute.
8. Repeat the process until the frames are completed and a global score is given.

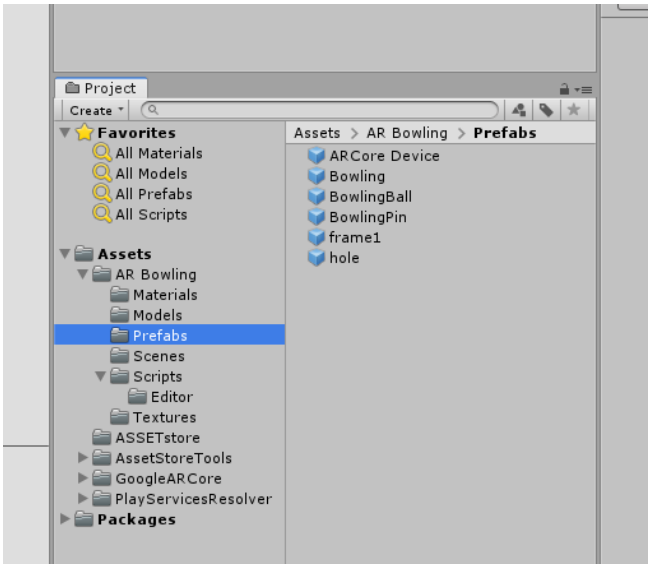
Full support pre-sales and post-sales at michael.soler.beatty@gmail.com. From cardboard buddies we pretend to give the best customer service, so we are available 24/7 for all doubts, errors in code development and potential modifications.

3.Tags

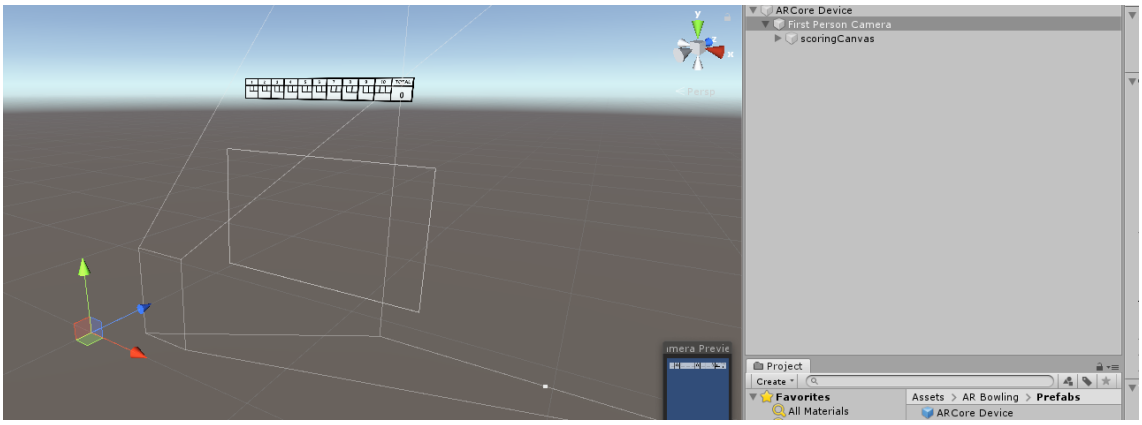
Make sure that these tags are in the editor mode:



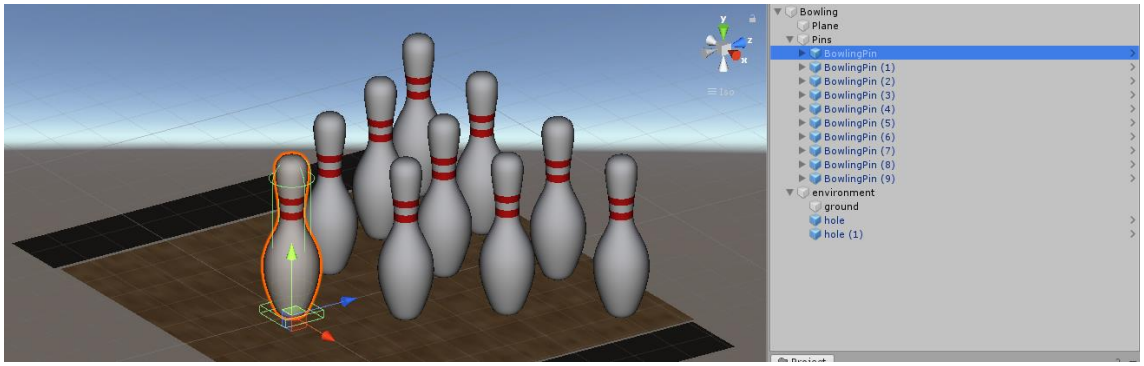
4.Prefabs



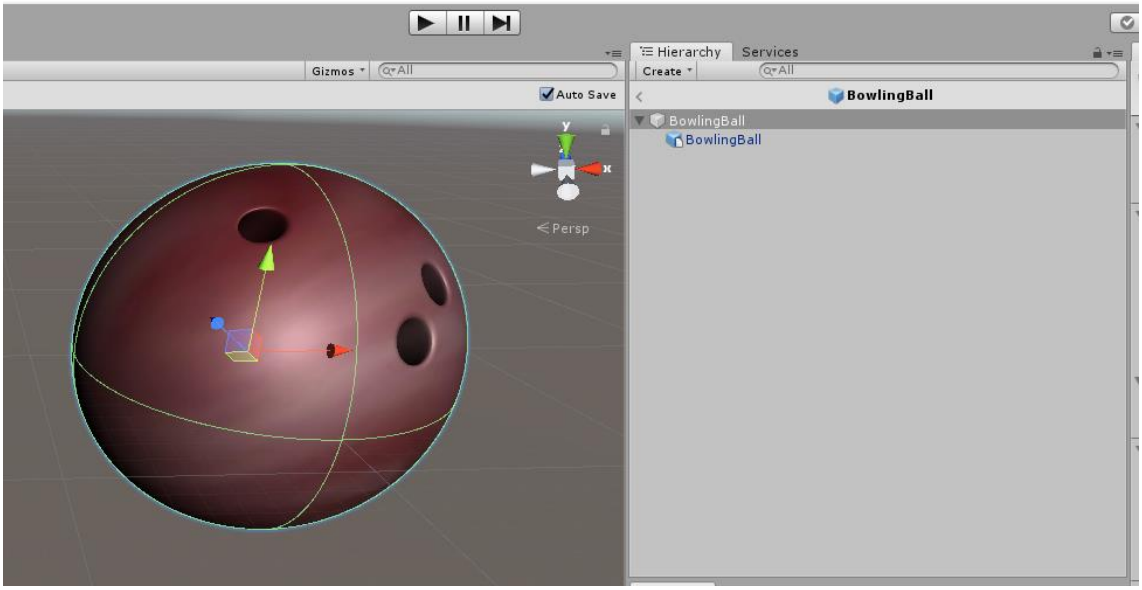
- **The ARcore device**:
  Contains the first-person camera and the canvas for scoring.



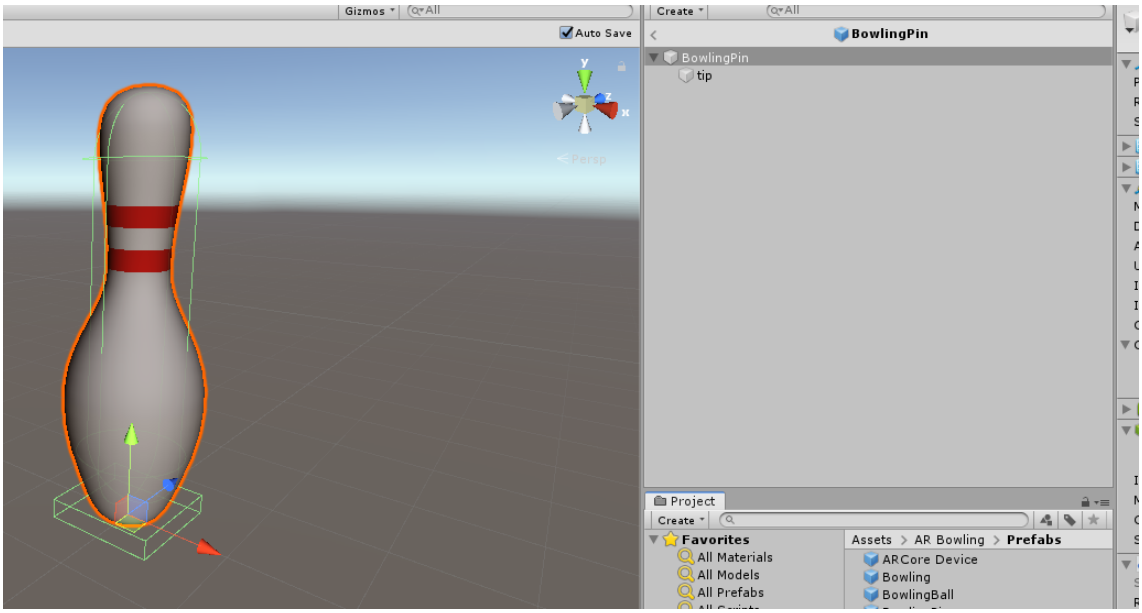
- **The Bowling:**
  This prefab contains the pins and the ground. It is the main gameobject in the scene that is enabled when the player selects an AR plane.



- **Bowling ball:**
  It is the ball used to score and knock the pins.

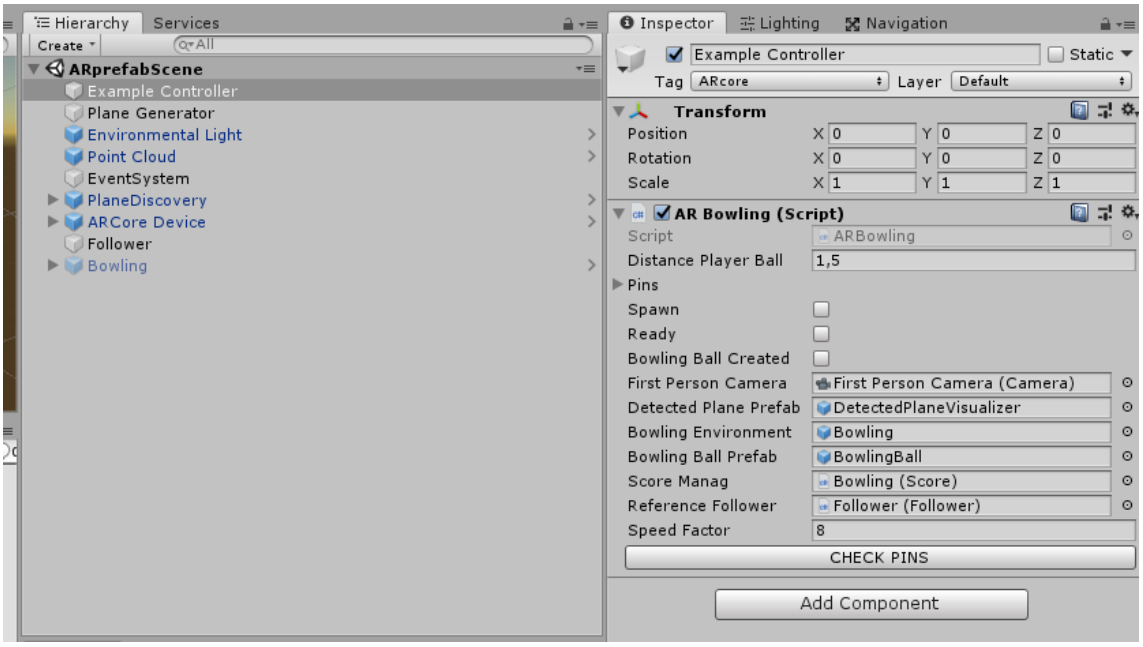


- **Bowling Pin:**
  It is the smallest prefab used inside the bowling prefab.



5.Scripting

The main script is the one used for interacting with the pins and is set in the example controller:



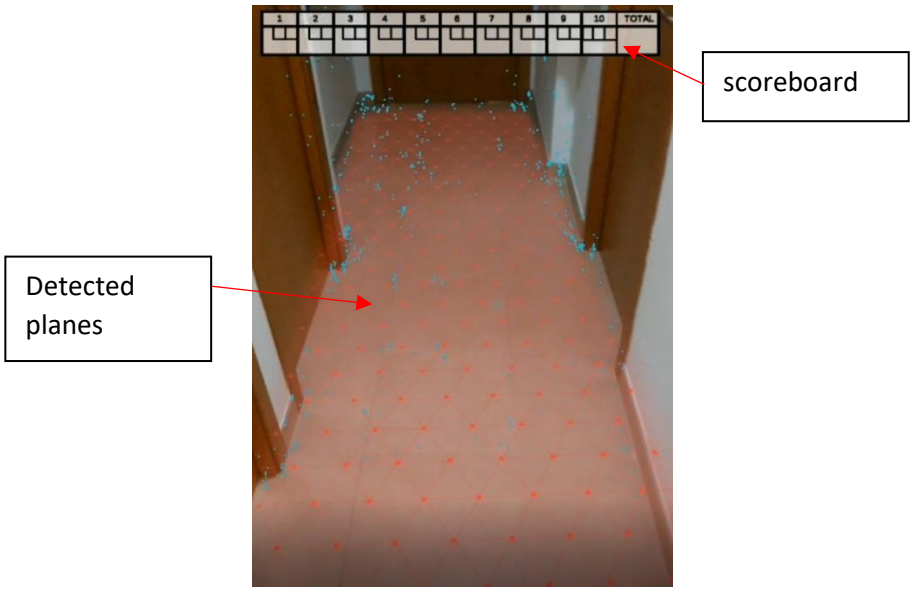
Distance player ball	Limits the movement of the ball when dragging
Pins	These are the pins on the scene
Spawn	It is used to enable the bowling environment only once.
Ready	Gives a certain time to prepare the movement of the ball
Bowling ball created	It is used to manage that only a ball es created per frame
Speed factor	The speed of the ball when moving.
CHECK PINS	This button is used for checking the state of the pins when they are knocked down.

```
////////////////////////////////////////////
            //this the bowling ball gameobject interaction
            ////////////////////////////////////////////
            if ((Input.touchCount == 1 && spawn == true && ready==true))
            {
                touch = Input.GetTouch(0);

                //use input of the touch for creating the position of the ball
use first line in editor

                //use raycast against the ground object to know where to move
the ball this works in a distance of 1m from the player only
                RaycastHit hit2;

                Ray ray = Camera.main.ScreenPointToRay(new
Vector2(touch.position.x, touch.position.y));
                if (Physics.Raycast(ray, out hit2, 300))
                {
                    if (hit2.collider.gameObject.tag == "ground")
                    {
                        pos = hit2.point + 0.01f *
hit2.collider.gameObject.transform.up;
                    }

                }




                if (bowlingBallCreated == false && touch.phase ==
TouchPhase.Began)
                {
                    bowlingBallCreated = true;

                    //instantiate the prefab

                    ballGo =GameObject.Instantiate(bowlingBallPrefab, pos,
Quaternion.Euler(Random.Range(0, 0), Random.Range(0, 0), Random.Range(0, 0)));
                    ballGo.transform.position = pos;

                    //disable collider when moving
                    ballGo.GetComponent<Collider>().enabled = false;

                    referenceFollower.t = ballGo.transform;
                    _rb = ballGo.GetComponent<Rigidbody>();
                    _rb.useGravity = false;

                }
                //while moving the ball
                else if (touch.phase == TouchPhase.Moved)
                {
                    if ((Camera.main.transform.position-pos).magnitude<
distancePlayerBall)
                    {

                        //move rigidbody
                        _rb.MovePosition(pos);
                    }
                }
                //when trhowing the ball
                else if (touch.phase == TouchPhase.Ended && ballGo != null)
                {
```

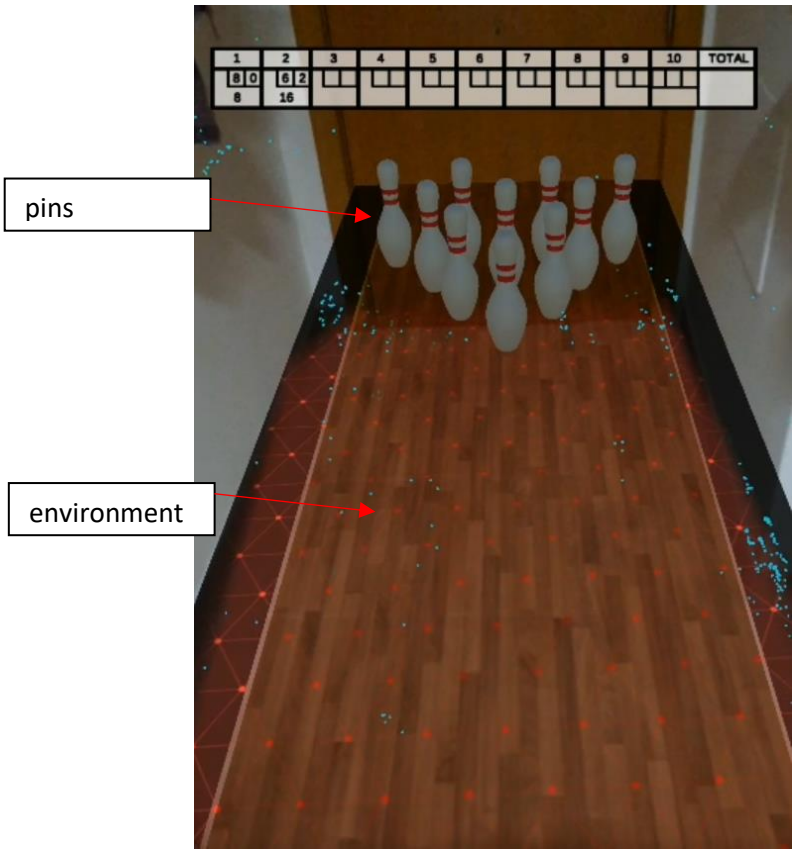
```
                    bowlingBallCreated = false;

                    //move rigidbody
                    _rb.MovePosition(pos);

                    //enable collider when release
                    ballGo.GetComponent<Collider>().enabled = true;
                    _rb.useGravity = true;

                    Vector3 speed = (pos -
referenceFollower.transform.position)* speedFactor;


                    _rb.velocity = speed;


                    referenceFollower.t = null;

                    _rb.useGravity = true;


                    ready = false;

                    Invoke("checkPins", 10);
                    Invoke("setReady", 13);
                    Destroy(ballGo, 10);
                    //ballGo.transform.GetComponent<Rigidbody>().velocity=
FirstPersonCamera.transform.forward*1;

                }


            }
```

6.WORKFLOW

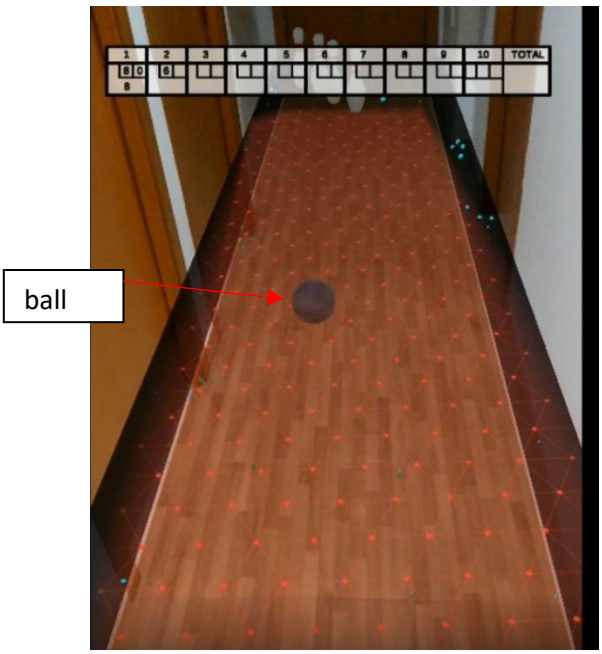
1. Detect planes process



1. Enable bowling gameobject by taping on screen



1. **Move ball by dragging on screen and releasing**



7.VIDEO TUTORIAL

We intend to give our customer the best service. To this aim, we upload tutorial videos of the asset always:

https://www.youtube.com/watch?v=3Ort82-A7RA