

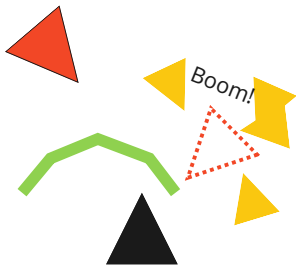
A1- Space Shooter Assignment - Journal

Joe Place

Mechanics

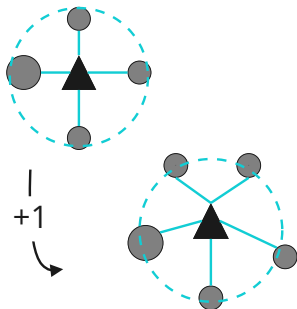
Projected shield:

The player can emit a shape like shield that can destroy objects that come into contact with it. The shield can only be active for a certain time and recharges when deactivated



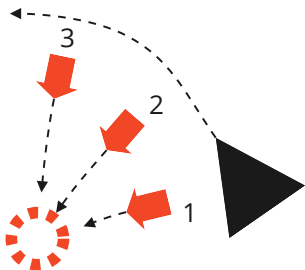
Asteroid Collector:

The player will be able to float near asteroids and collect them. Each asteroid collected will be connected to a point floating around the player and will move with the player while the collector is activated. Every additional asteroid will increase the amount of points around the player keeping the asteroids in a ring like configuration.



Broadside missiles:

The player can press a button to fire missiles from certain sides of their ship that will continue forward to a fixed focal point then continue. The missiles will fire separately overtime while going to the same point. The missiles will explode after some time.

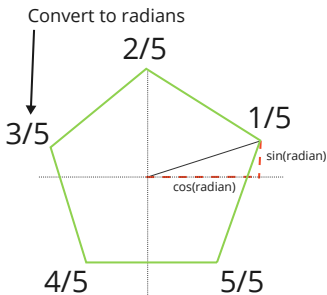


Session 1: 0:00 - 2:35 - Shield



Pressing the shield button will generated a shield in from of the player. Anything that touches or enters within a certain radius of the shield will be destroyed.

The generation will use draw lines to vertices based around a origin point. The detection will create pseudo colliders with it checking enemy position, expanding out a radius and if that point enters close to the shields points it will be destroyed.



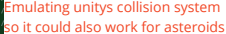
Math:

To draw the half circle or projected shield, the points are generated by having a fraction of where the point would be on a circle and multiplying it by pi, riving a radian value. By converting the radian to degrees I can use cos and sin to plot the vector in world space. This point will be added to a list of transforms, in a list format the fraction is helpful as it can iterate through a loop and take the fraction of the place in the loop to the overall amount of points.

For enemy detection, a for loop would take the enemies position and get the magnitude of the enemies position to each of the points in the "projected shield". If this magnitude exceeds a small value to be similar to contact the object is destroyed.

Video I used for reference for drawing and generating the points
<https://www.youtube.com/watch?v=DdAfwHYNFOE>

- Having a specific reference results in null value on destruction

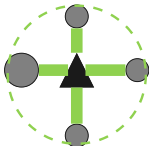


This could allow me to reference aspect of the collided like the destroy condition

Currently the shape fails to go over 8 steps which is fine. I changed the radian calc to restrict it to a half circle. This was confusing for me as when I reduced it to half it did restrain but increasing it to 4 showed little change as if it was clamped. Upon inspection reducing it to half didn't do much. I changed it to simply draw and the loop to be restricted to only half of the joints

I tried to emulate how the collider system works as I have experience with colliders but wanted to avoid them for this project. The pseudo collision relies on having a predetermined object to reference so when calling the function I would still pass it the enemy in script. This is an issue as if this doesn't work nothing else did either. For the sake of scope and time constraints I don't think I can make a general controller to pass in all enemies spawned and asteroids collided with

Session 2: 2:35 - 4:40 - Asteroid Collector



Asteroid grabber will take any asteroid within a radius and attach it to a vertice around the player, it will then move with the player and the player can release the asteroid using a list to track which point was attached first.

Math:

It will use the same point generation method as the Shield but the detection has to be handled for every asteroid in the scene to each point generated.


Assignment:

When an asteroid exceeds a limit in its magnitude to a point it will be detected in a for loop and share the position vector of a point on the circle. The index will then be excluded from the for loop (somehow). My idea for the exclusion could be another separate list or array.

```

0; i < points.Count / 2 + 1; i++)
    Didnt see this till reviewing document
j = 0; i < asteroids.Count; i++)
    Vector3 asteroidTarget = asteroids[j].transform.position - points[j];
    float magnitude = Mathf.Sqrt((asteroidTarget.x * asteroidTarget.x) + (asteroidTarget.y * asteroidTarget.y));
    if (magnitude < 6f)
        asteroids[j].transform.position = points[j];

```



Asteroids are defaulting to the point[0]

The Asteroids are detected but excluded to the incrementing numbers with no exclusion. Some asteroids (any about the circle limit) are unable to be detected

```

void AsteroidCollection()
{
    //all I want is for an asteroid to be collected if it comes within the detection distance
    //Debug.DrawLine(points[0], asteroidTransforms[0].position);
    Vector3 target = asteroidTransforms[0].position - transform.position;
    float magnitude = Mathf.Sqrt((target.x * target.x) + (target.y * target.y));
    //problem seems to be with the asteroid transforms, why?

    Color radarColor = Color.white;
    Debug.DrawLine(transform.position, asteroidTransforms[0].position, radarColor);
    if (magnitude < asteroidDetectionDistance)
    {
        //Assigning the asteroid to the point
        //Test for assignment is reliant on
        //asteroids index = points index
        //maybe I should have it so it is within detection distance
        asteroidTransforms[0].position = points[0];
        //Now this works but it does not lock out the different asteroids
        //so I need a way to set a number to be false or not
        //as well if a point is set as true it then gains another asteroid
    }
    //Experimenting with collection limit
    //and where to break the for loop
    //collectionLimit while it runs
    for (int i = 0; i < asteroidTransforms.Count; i++)
    {
        if (i > collectionLimit) { break; }
    }
    //However collectionLimit was used in many circumstances till it became too difficult to debug
    //especially the expansive intention for the points
    Circle(collectionLimit, 5f, Color.blue, 1);
}

```

```
public List<Asteroid> asteroids;
```

Asteroid objects are only the class so gameobject functions are convoluted

```

public float moveSpeed;
public float arrivalDistance;
public float maxFloatDistance;

public Vector3 position;

public bool isAttached;

```

Exploring the asteroid having booleans but seems like it needs a global manager to be sustainable

Integrating the idea has brought issues with how the loop is handled with it sometimes ignoring some indexes. As well it would only assign the equivalent asteroid to its equivalent point which there were a limited amount of points often returning an error.

I have started exploring options to avoid and solidify the asteroid assignment methods but nothing seems to work properly. I tried having collection limit and have that increase the amount of asteroids that could be assigned but learnt that you cannot change a for loops range during runtime (This took too long to learn).

I am starting to think of a system that can reference a property on the asteroid like a boolean so that the asteroids can exclude themselves when attached but I would prefer to alter the asteroids for scope reasons.

I tried having the asteroids as specific object to reference this but without having a global manager to specify which parts of the asteroid objects could be referenced the asteroid objects would only reference the class. This would exclude the transform and communication between the objects seems too complicated for the purpose

```

for (int i = 0; i < points.Count / 2 + 1; i++)
{
    for (int j = 0; i < asteroids.Count; i++)
    {
        if (asteroids[j].transform.position != points[j])
        {
            //currently it is detecting the first one and
            Vector3 asteroidTarget = asteroids[j].transform
            float magnitude = Mathf.Sqrt((asteroidTarget.x
            if (magnitude < 6f)
            {
                asteroids[j].transform.position = points[j]
            }
        }
    }
}

```

Hard to loop at

This was trying to exclude if the transforms and the points were equivalent. Did not work as the chance of them being equal is so low it would never check at the precise moment. As well the implied movement would cause too many issues with this. All the nested loops look ugly I will try to see if I can shrink some

Session 3: 4:40 - 6:00 - Asteroid Collector Cont.

```

//can i not increase a for loops limit while it runs
//i need it to somehow ignore an asteroid that has already been caught
for (int i = 0; i < asteroidTransforms.Count; i++)
{
    if (checkAvailableSpaces() == -1) { break; }
    Color radarColor = Color.white;
    Vector3 Target = asteroidTransforms[i].position - transform.position;
    float magnitude = Mathf.Sqrt((Target.x * Target.x) + (Target.y * Target.y));
    Debug.DrawLine(transform.position, asteroidTransforms[i].position, radarColor);
    if (magnitude < asteroidDetectionDistance)
    {
        //i need to delay this so it checks it once and ignores said asteroid
        radarColor = Color.red;
        asteroidTransforms[i].position = points[checkAvailableSpaces()];
        IsPointOccupied[checkAvailableSpaces()] = true;
    }
}
Circle(collectionLimit, 1.5f, Color.blue, 1);

```

A little issue I had was involving collectionlimit and being unable to increase the duration of a list during run time which would break the program. Took too much time trying to figure out that I cannot modify loop ranges during runtime.

```

public int checkAvailableSpaces()
{
    for (int i = 0; i < IsPointOccupied.Count; i++)
    {
        if (IsPointOccupied[i] == 0) { print($"not occupied point{i}"); return i; }
    }
    return -1;
}

```

Function returning available index

With the belief of needing another extension to track asteroids I tried adding a list of booleans to track the availability of the slots. This does remove the intention of the asteroids shape expanding with more asteroids but I need something that works.

I then had a function that would iterate and return the first open space giving an index value that could then be used to assign the points value. I also had a means to break the additional detection if it was full.

I have thoughts on expanding it to receive any list given but I may not have the need for such a function. This would also not return null so I set the base return if full to -1.

Having another list did not work as juggling which index was available and assigning it to that specific one within a majority happened too quickly resulting in it assigning one asteroid to all available slots. And became a logistic nightmare

Then I changed the extra list to save the index of the asteroid it would come in contact with to an available spot in `IsPointOccupied`. `CollectionLimit` would work as a manual index where first asteroid goes to point one and so on (a method like this seemed more approachable for my skill then attach to closest point). This wouldnt work as again some asteroids were locked out due to loop restrictions as well I have it handled so that the asteroid would not continue to update its position.


```
if (collectionLimit >3) { break; }
Color radarColor = Color.white;
Vector3 Target = asteroidTransforms[i].position - transform.position;
float magnitude = Mathf.Sqrt((Target.x * Target.x) + (Target.y * Target.y));

Debug.DrawLine(transform.position, asteroidTransforms[i].position, radarColor);
if (magnitude < asteroidDetectionDistance)
{
    //I need to delay this so it checks it once and ignores sa

    radarColor = Color.red;
    //this will retain the index of the detected asteroid
    i = IsPointOccupied[checkAvailableSpaces()];
    asteroidTransforms[i].position = points[collectionLimit];
    collectionLimit++;
}
```

It wouldnt update because the assignment came after the break

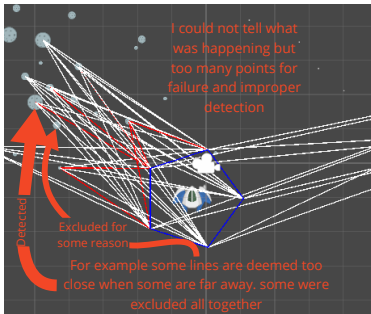
Movement issues



I changed it to save the index to try and have the collected asteroids update outside of the function. However it would often default to a different asteroid then the one touched as the for loop goes on an index basis but the `IsPointsOccupied` still has legacy numbers that weren't erased

```
if (IsPointOccupied[i] != 0)
{
    asteroidTransforms[IsPointOccupied[i]].position = points[i];
}
```

I tried having the positions update in update with the specific index but the lists were not set up to ignore certain values like 0. So during runtime certain asteroids would attach to different points and `IsPointOccupied` also didnt save the index correctly for there was a issue with the for loop detection



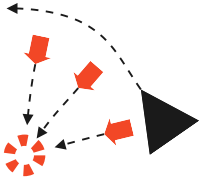
As the system became more expansive, the handling of multiple objects became overwhelming to track with the points not lining up and the indexes not orienting properly with what was detected as the points were either checked in a different loop or the same which would either restrict which asteroids could be detected or default all asteroids to the first point.

A big point of failure was every asteroid had to be handled manually so although some were missed in the list the ones in the list were still sometimes missed completely as with the saved index system excluding some.



In light of the egregious time sink for one mechanic I opted to have remove the multiple asteroids entirely to attach one asteroid and release after the player stops pressing the button. as well since the ship does not move the shield will be a whole shape to avoid rotating the shield and the points calculation.

Session 4: 6:00 - 7:00~ - Broadside missiles



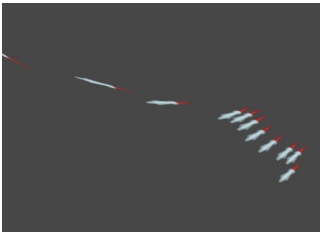
Broadside missiles will spawn missiles over a small increment of time all angled toward a offset to the ship at the time of spawning.

The missiles would be spawned using a coroutine that would feed the instantiation an updated direction and rotation with a number of spawns and interval between the burst.

The missiles rotation would be a direction to a fixed location on the side of the ship, where using the rotation I can use the quaternion.LookRotation to have the quaternion rotation of what the value would be for when the missiles were spawned.

Unity documentation for the rotation:

<https://docs.unity3d.com/ScriptReference/Quaternion.LookRotation.html>



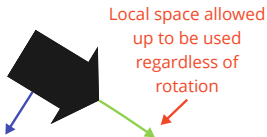
Using LookRotation was met with an immediate issue with the second argument of the function. the second argument uses a vector to determine up and orientate the object. This became an issue as the object were spawned and then oriented to have the sprites local space (the missiles +y be forward) would become oriented in the z value having them become "invisible" as they were oriented away from the camera

This as well as the constant update of the players transform and its offset caused the direction to be inconsistent. As well using LookRotation would use global direction compared to the local direction of where it was spawned.

```
Quaternion MissilesDirection()
{
    //missiles can have their own script that moves them when spawned at a c
    //z axis is forward
    //need direction w
    //give it a Local:
    //Angle is the tangent between the x and y giving a float
    //quaternion rotation angleAxis angle - 90)
    Vector3 direction = Camera.main.ScreenToRay(Input.mousePosition).direction;
    direction = direction.normalized;
    float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;
    Quaternion rotation = Quaternion.AngleAxis(angle - 90, Vector3.forward);
    return rotation;
}
```

legacy code for lookRotation

Taking the direction to the mouse we can get the angle and apply it to the missile



Another issue with the original design was that the ship would rotate as it moved but the player does not rotate at all so the broadside nature of the mechanic became flawed and unintuitive for the game. So I changed it to be directed at the mouse. This had the direction work properly but the missile could not move and instantiate could not simply give it a force or speed.

```
Translate(Vector3.up * 10 * Time.deltaTime, Space.Self);
```

Space/self means the Vector.up can be "forward" even when rotated

Another issue with the original design was that the ship would rotate as it moved but the player does not rotate at all so the broadside nature of the mechanic became flawed and unintuitive for the game. So I changed it to be directed at the mouse. This had the direction work properly but the missile could not move and instantiate could not simply give it a force or speed.

So I gave the missile a script that would move it. The issue with this was that the movement method of the player is dependant on world space so the missile would not move "forward" in the intended direction. My first idea was to have the player pass the direction vector to the missile and have it simply move said vector each frame. However, I found the transform.translate function which moves it in a direction and designated which space it would use for the calculation

```
IEnumerator BroadsideMissiles(int numOfMissiles, float ShotInterval)
{
    for (int i = 0; i < numOfMissiles; i++)
    {
        isShooting = true;
        Instantiate(missile, transform.position, MissilesDirection());
        yield return new WaitForSeconds(ShotInterval);
    }
    isShooting = false;
}
```

Then for the sake of design and simple integration I made the coroutine take different public parameters to allow variation in the missiles burst function. An emergent feature of the mouse direction methodology is that it doesn't take when it was clicked it takes current position which can have the player lead their target.

What I learned:

I miss the collision system in unity. I have some experience with colliders and while coding specifically the asteroid collisions I could theory craft how to do it with colliders and believe it would be trivial. However having to design a pseudo collision system became a nightmare for someone that does not have much experience with coding and was negligent to create a global manager. I do however find it a worthwhile challenge to program within the given constraints of the system given and not completely change the system. With that approach I did learn to adapt with certain systems and try to weave together solutions. But I do now think having the asteroids handled with a global manager that could feed and select information would have been helpful for the project and probably allowed more expansive creativity instead of mass reductions in functionality.

I do believe I have a more solidified understanding of how a collision system works as well as how to manipulate objects with rotations and utilize the unit circle in more adaptable ways. I do think journaling was useful in understanding the scope of the project but I do not feel I fully utilized the journaling moments as a retrospective as well as a break to search for references. Going forward I do think I will create more technical documents or journals to reference solutions and techniques for later as there were moments in the project where I forgot certain formatting.