Robes, Wrinkles and Magical Evils

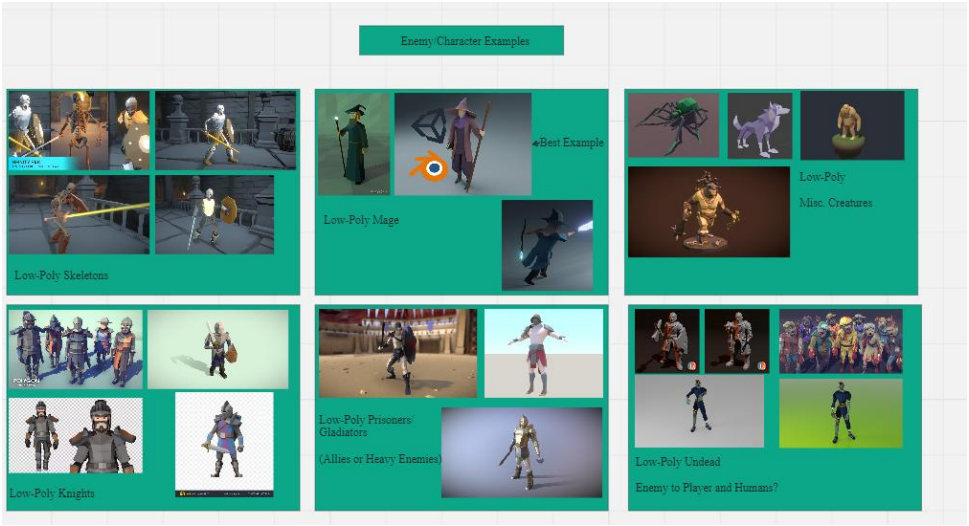
Game Overview

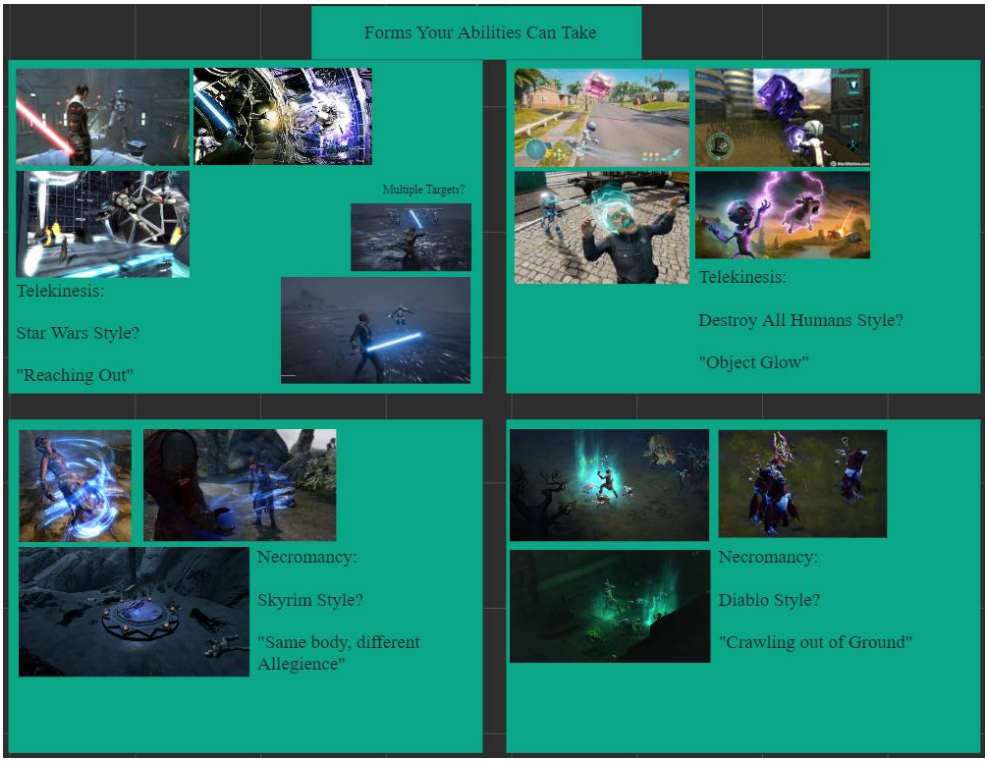
Robes, Wrinkles and Magical Evils is a physics based first-person dungeon crawler, it is centred around manipulating physics and using sorcery to progress through rooms and up a large tower. The game is designed with the intention of combining Gang Beasts' physics with Mike Magnola's Hellboy art style. To this extent, flat colouring and a more comic book centred style was focused upon for the art direction.

Figure 1 in-engine art.

Research

In order to gain a good grasp of what we needed, a lot of my time was spent researching different themes and elements from multiple genres of games. Much of my research focussed upon areas that were fantasy driven as I took a keen interest in the way magic typically operated within games mechanically. Following on from this, I also took the time to compile several simple character mood boards and potential ability-based mood boards.

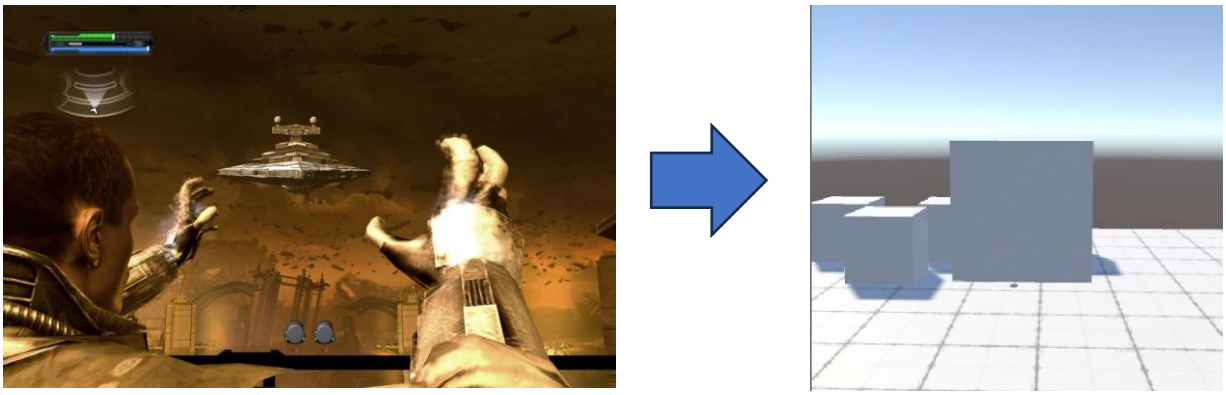




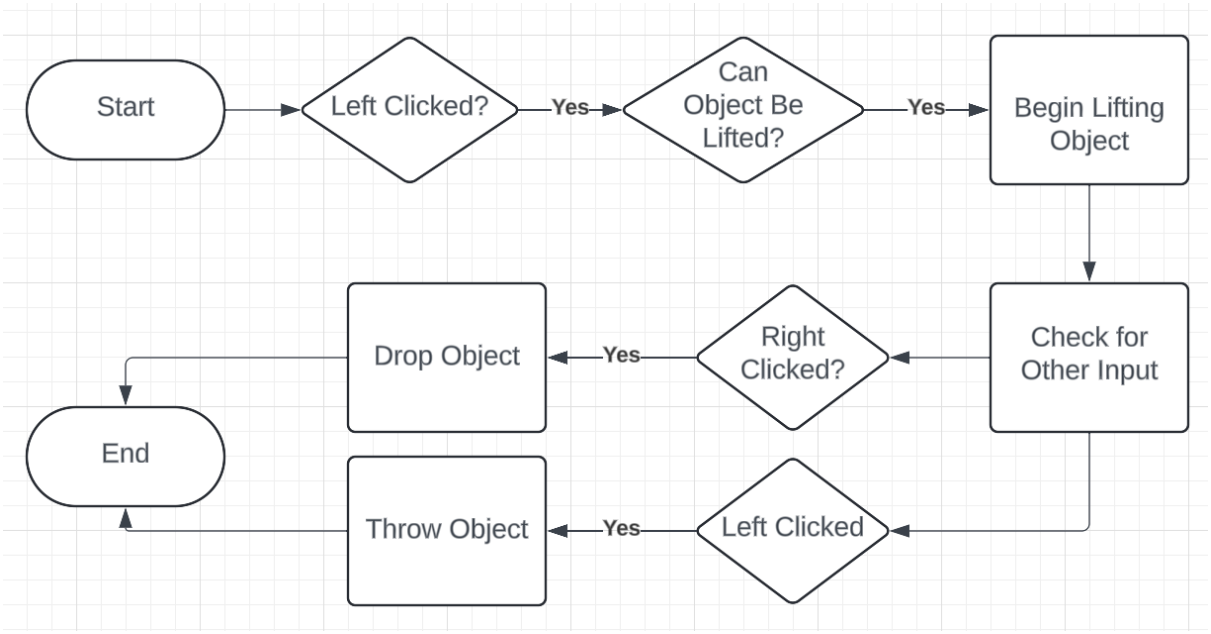
Early Development

Telekinesis

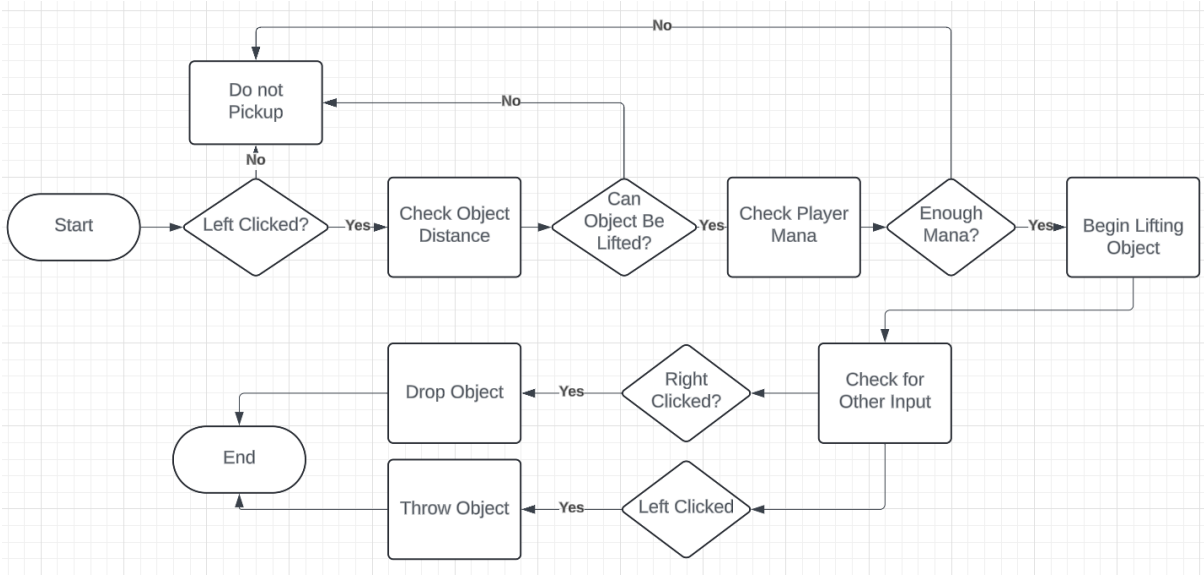
During the development stage, my first step was to begin to work on a prototype for the “Telekinesis” ability. This involved looking toward lots of references for how different games handled their own versions of this skill. The key point of reference became Star Wars: The Force Unleashed for its physics-based force powers



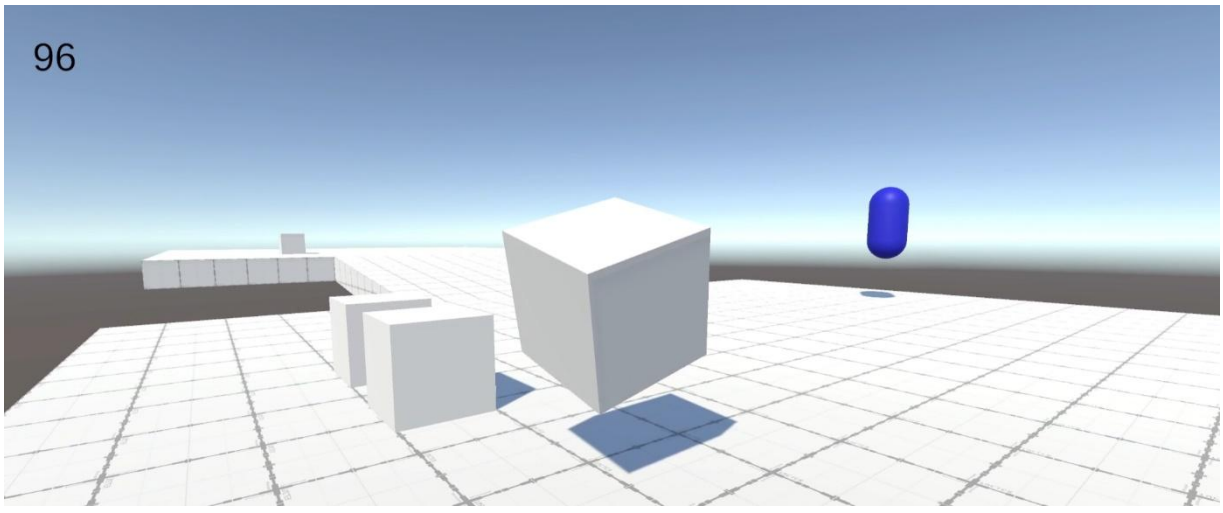
As shown from the first example, the player could lift objects which would then follow them around based on where they looked within the level. Following on from feedback via team members, the player was then given the ability to throw objects and drop them which I planned for using a simple flowchart



Expanding upon this further, before putting the work in practice I developed the flow chart to include more parameters such as a maximum distance to pick objects up and a “mana” value that drops the object if it reaches zero. This was done for the sake of future proofing balance.

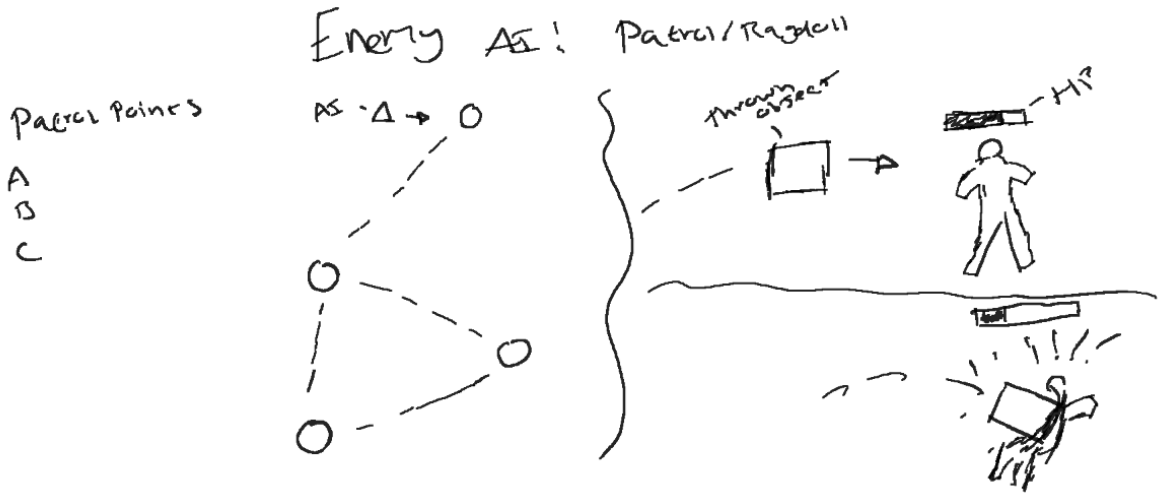


As shown below, I created a working prototype for the players “Telekinesis” ability which would be later reworked and adjusted based on feedback from team members and play testers.



Enemy AI

My next task was developing a very simple AI to use as both a test dummy and framework to build from. Prior to any programming, I sketched out two basic behaviours I wanted to create to make sure I could pace my work. These would be known as the ragdoll and patrol behaviours.

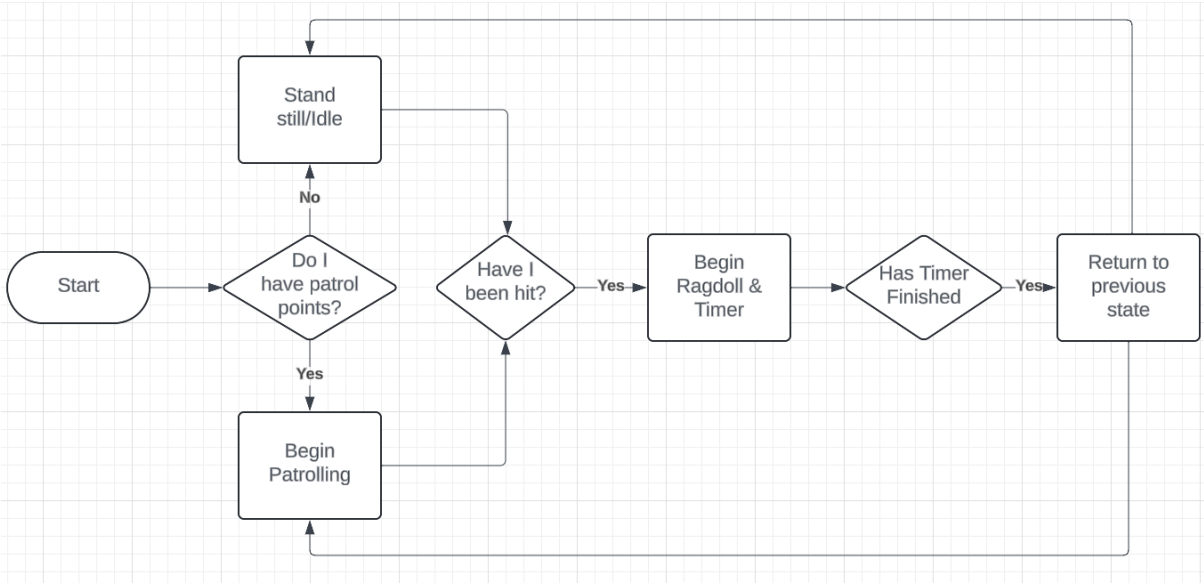


In order to create a working brain for the characters, I began creating a state machine framework. The Statemachine script determined what state the character would be in and what it would do whilst the state script contained the basic functions of each state.

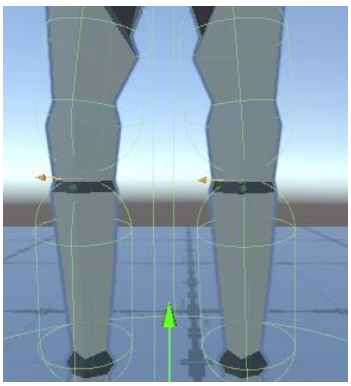
```
public abstract class State
{
    17 references
    public abstract void Enter();

    17 references
    public abstract void Tick(float deltaTime);

    17 references
    public abstract void Exit();

}
```

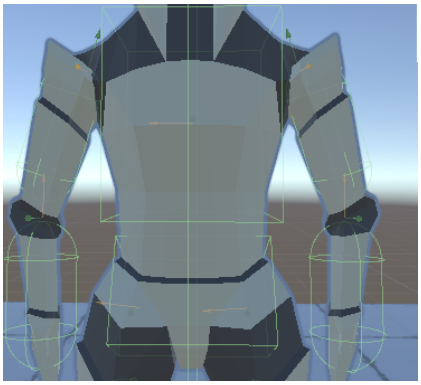
```
public abstract class StateMachine : MonoBehaviour
{
    private State currentState;

    26 references
    public void SwitchState(State newState)
    {
        currentState?.Exit();
        currentState = newState;
        currentState?.Enter();
    }

    Unity Message | 0 references
    private void Update()
    {
        currentState?.Tick(Time.deltaTime);
    }
}
```

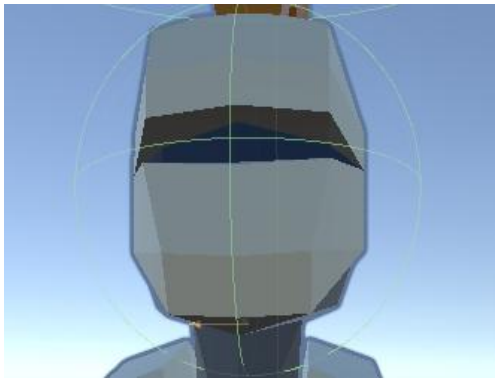
Following on from this, similar in how I did the Telekinesis ability, I designed a small flowchart which helped to visualise my ideas for not only myself but others within my team.



Looking towards implementation, I made use of a character created by a team member and gave each limb physics-based properties. I then gave it the brain I had worked on in script and began testing. The most difficult portion of this was making sure colliders for the character fit correctly as shown below, this meant manually adjusting the bounding boxes in-engine.







After the character was adjusted properly, the last part of this section was to get the character to ragdoll in order to simulate the player “Killing” it. In practice, this meant turning off the animation module and turning on the collider physics. This was met with some very positive feedback from the team and so I practiced different methods.

During this testing process I discovered some unique physics based “features” when the enemy entered the ragdoll state. This led to a lot of laughter from the team and created several ideas that could be incorporated into the comedic element within the game.