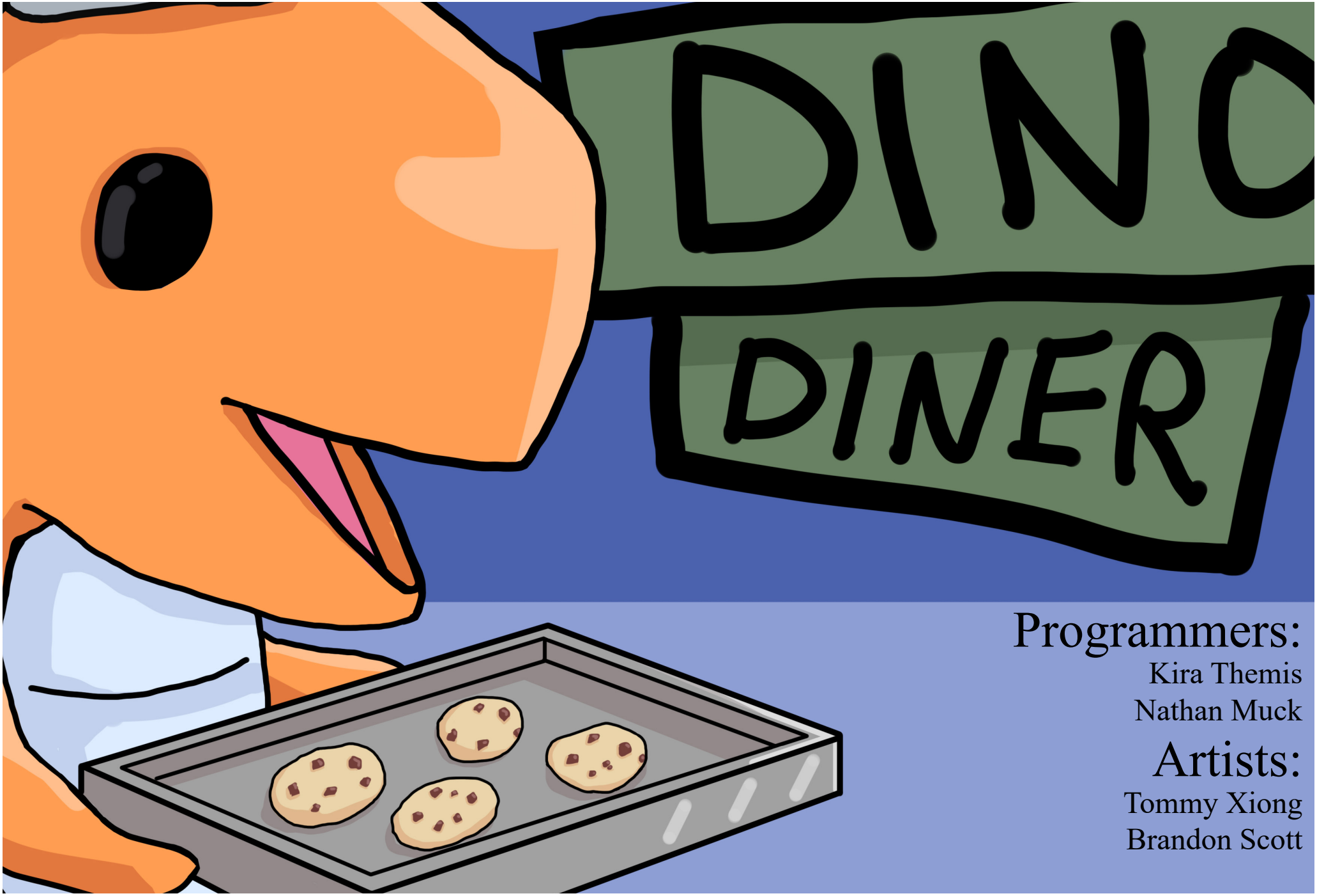


Programmers:

Kira Themis

Nathan Muck

Artists:

Tommy Xiong

Brandon Scott

Elevator Pitch

Imagine the love and joy of the customers in ‘Tapper’ mixed with the raw power and speed of ‘Overcooked.’ What we are left with is the next upcoming classic simply known as ‘Dino Diner!’ You play as a cute T-Rex just trying his darnedest to make cookies for his adoring public, who are also dinosaurs!

Target Demographic

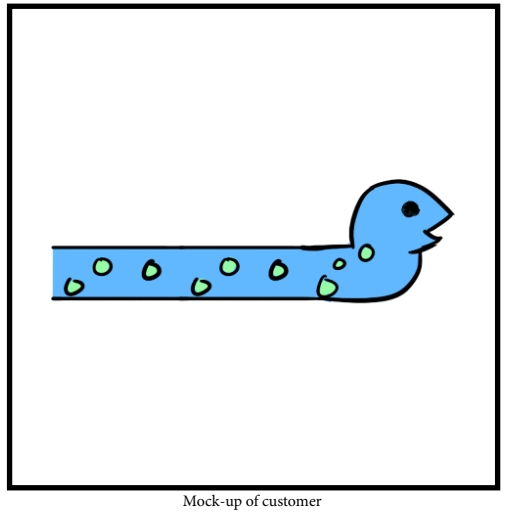
We wanted to make this game with the term ‘casual’ in mind. So it’s safe to say that we created this for casual players but we also wanted anyone who played it to feel like they’re having fun but not over-whelmed with rules or content.

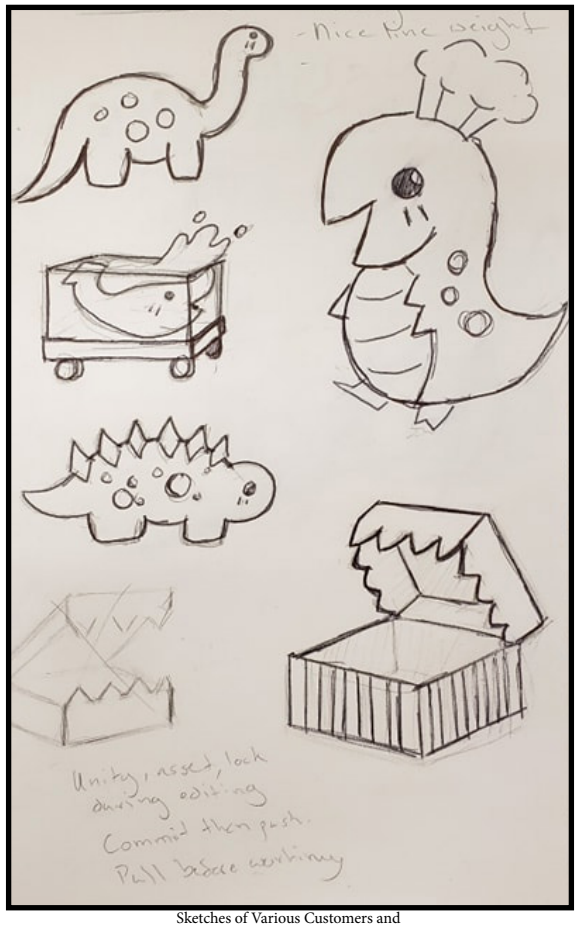
The concept is simple, the execution is fun. This truly is a game that can be considered ‘E’ for everyone.

Concept Art

We wanted to make the characters cute and fun so that everyone could love them. Here we can see early iterations for the main character as well as some mock-ups for customer characters and objects.

Main Character Sketches


Mock-up of customer



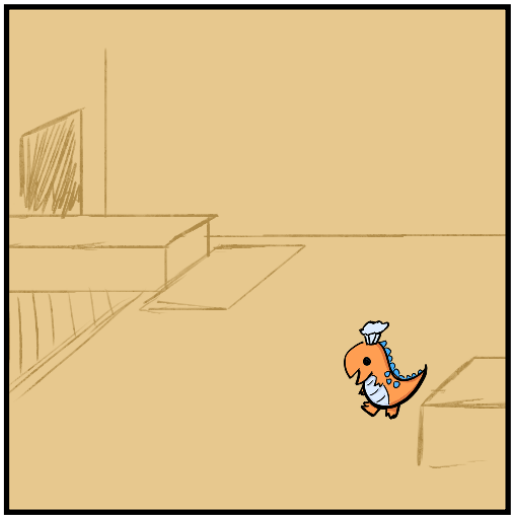
Sketches of Various Customers and Main Character

Concept Art

After rounding out the characters, we began to develop a level layout and a basic movement system for the player character on top of that.

The bit of visualization proved extremely useful in moving forward as we had a clear vision of what needed to be done.

Early Mock-ups of level design



Outline of Experience

WEEK ONE:

- Brainstorm ideas for a cookie-based game, narrowed down to two ideas
- Quickly prototype those ideas to see what's inherently 'fun'
- End on one concept to work with, 'Dino Diner'

WEEK TWO:

- **Artists**:

o Begin concepting characters and level design. Working through various iterations before settling.

- **Programmers:**

o Begin creating systems for the player character and the customers, respectively. Refining the code as they go along.

WEEK THREE:

- **Artist:**

o Refine designs and prepare for filler material to liven up scene. Make corrections involving perspective and readability.

- **Programmers:**

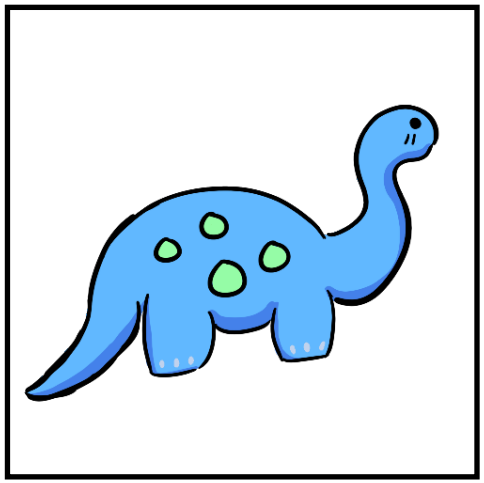
o Continue testing code and creating new code for new problems that arise.

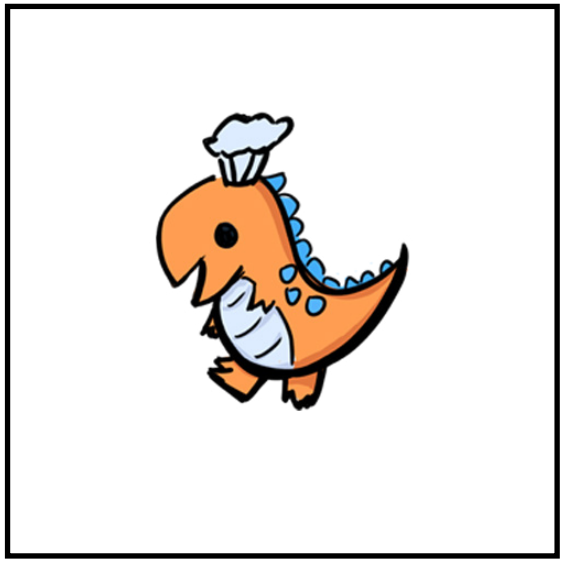
Inspiration

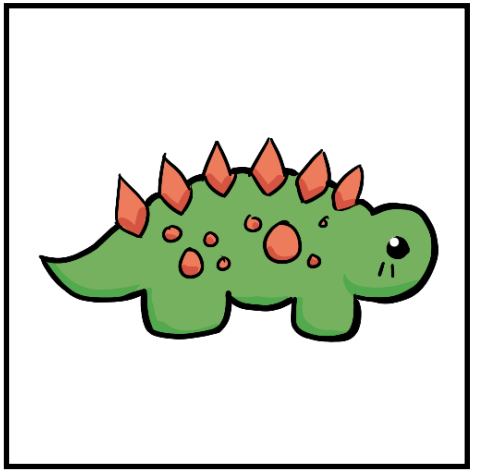
Heavily inspired by Tina's own T-Rex Cookie Kitchen.

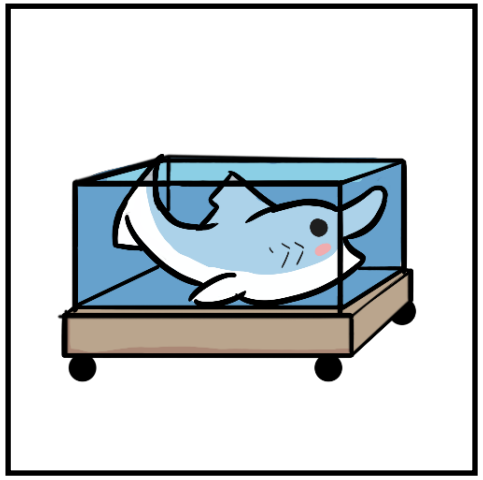
We were also inspired by games such as the classic 'Tapper' along with the other classic 'Overcooked'

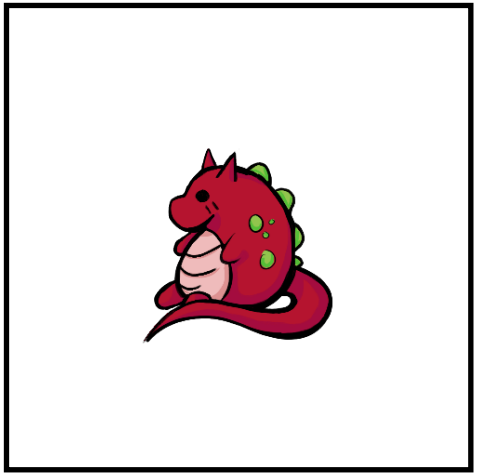
Game Artwork



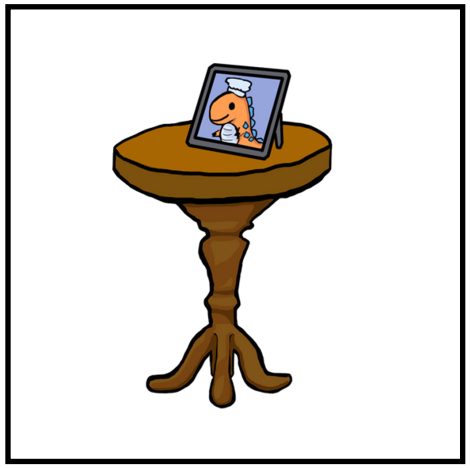


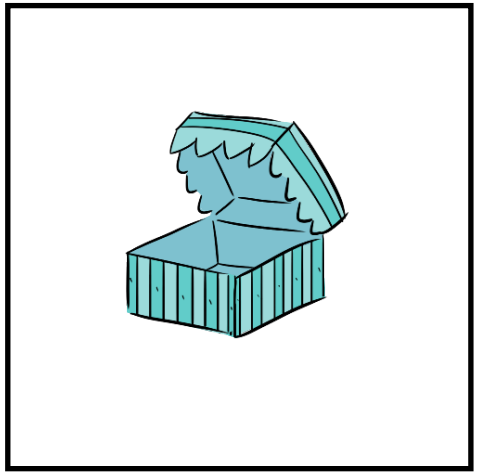


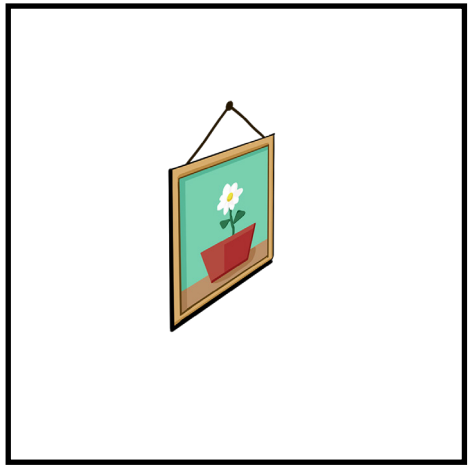


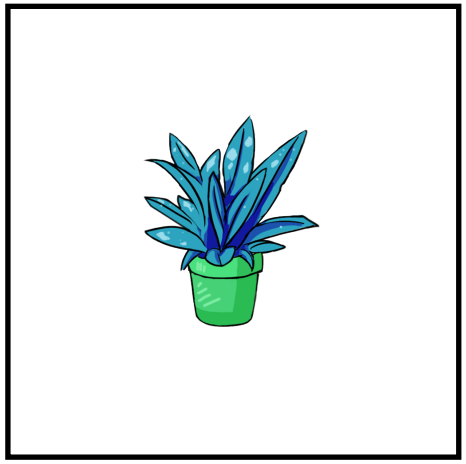


Game Artwork

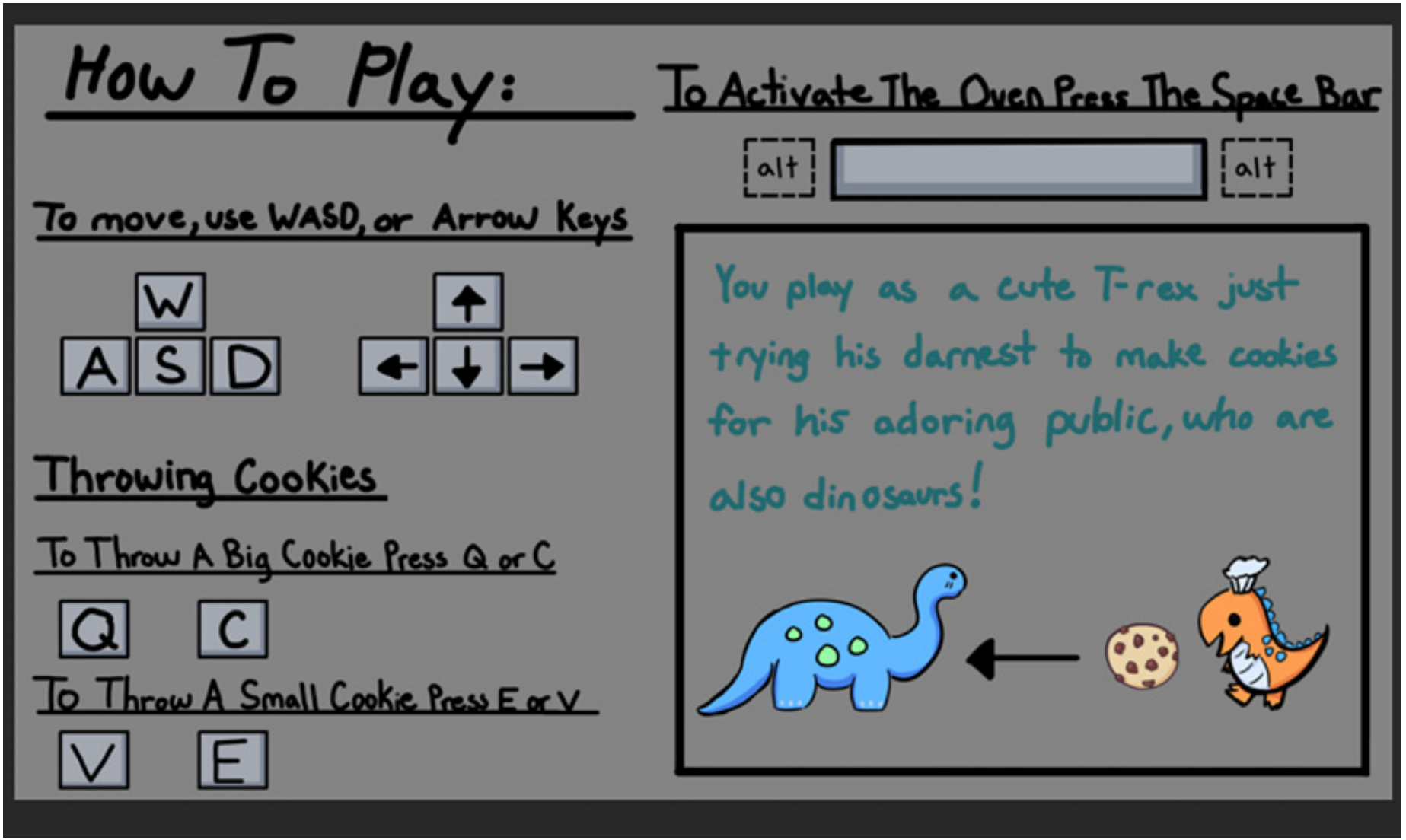






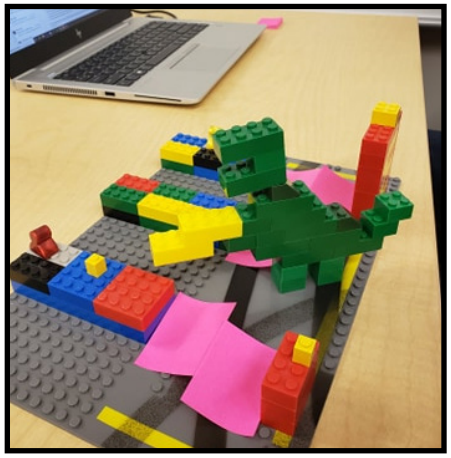


Game Artwork

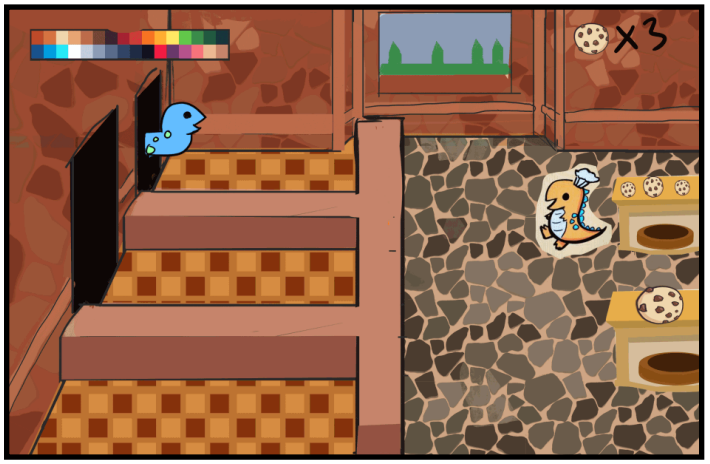


Evidence of Iteration

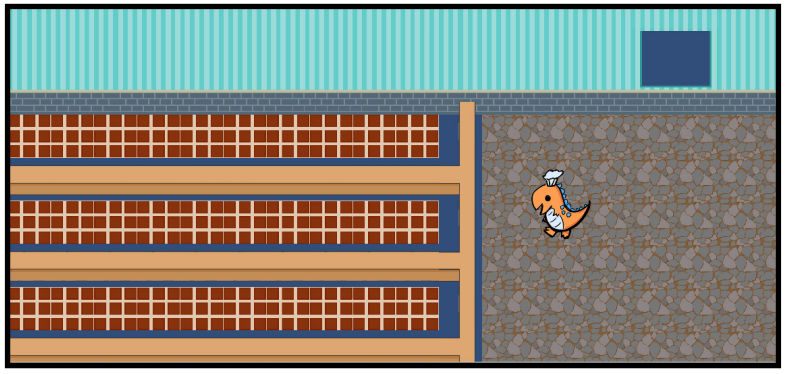
Prototype start



Early Mock-up



Tile Map Test



Final Product

Code Snippets

```
public void bakeCommand()
{
    //this can be called elsewhere to start the baking.
    timeLeft = bakeSpeed;
    baking = true;
}

4 references
public void resetVars()
{
    //resets the variables so that the baking can be done again
    baking = false;
    done = false;
    burning = false;
    burned = false;
    timeLeft = 0;
    linkedHandle.SetActive(false);

}
```

```
    {
        //activates baking and counts down the time, and if time runs out a second time the cookies be
        linkedHandle.SetActive(true);
        print("timeleft= "+ timeLeft + " done=" + done + " baking=" + baking + " burned=" + burned);
        //counts down the time
        timeLeft -= Time.deltaTime;
        //this if statement is what determines the first countdown for the cookies to be cooked.
        if (timeLeft <= 0 )
        {
            done = true;
            burning = true;
        }
        //this statement determines if the cookies become burnt
        if (timeLeft < -bakeSpeed)
        {
            done = false;
            burned = true;
        }
    }
}

2 references
```

```
            gameManager.setText();
            smallOven.resetVars();
        }
        if (smallOven.burned)
        {
            //if the cookies are burned, reset the variables and don't
            smallOven.resetVars();
        }
    }

    else if (this.transform.position == locations[3])
    {
        if (!bigOven.baking)
        {//starts baking
            bigOven.bakeCommand();
        }
        if (bigOven.done)
        {
            //collects big cookies and resets variables
            gameManager.bigCookies += 4;
            gameManager.setText();
            bigOven.resetVars();
        }
        if (bigOven.burned)
        {
            //if cookies are burned then reset variables and don't give
```

```
//these if statements handle the looping if the player hits the bottom of the screen
if(pos == 5)
{
    pos = 1;
}
if(pos ==6)
{
    pos = 0;
}
if (pos == -1)
{
    pos = 3;
}
if (pos == -2)
{
    pos = 4;
}
//moves the player to a new location
rb2D.position = locations[pos];
//these flip the player model if they are on the left or right side
if (pos == 1 || pos == 3)
{
    sprite.flipX = true;
}
if (pos == 0 || pos == 2 || pos == 4)
{
    sprite.flipX = false;
}
```

```
//this function gets input either from wasd or the arrow keys. Up and down increase/decrease by 2; left and right increase/decrease by 1
if (Input.GetKeyDown(KeyCode.W) || Input.GetKeyDown(KeyCode.UpArrow))
{
    pos -= 2;

}
if (Input.GetKeyDown(KeyCode.S) || Input.GetKeyDown(KeyCode.DownArrow))
{
    pos += 2;
}
if (Input.GetKeyDown(KeyCode.A) || Input.GetKeyDown(KeyCode.LeftArrow))
{
    pos -= 1;
    if (pos == -1)
    {
        pos = 4;
    }
}
if (Input.GetKeyDown(KeyCode.D) || Input.GetKeyDown(KeyCode.RightArrow))
{
    pos += 1;
    if (pos == 5)
    {
        pos = 0;
    }
}
```

```
locations[1].Set(50.94f, -0.18f, 0f);//small oven
locations[3].Set(52.15f, -6.36f, 0f);//big oven
locations[0].Set(42.01f, 2.09f, 0f);//row 1
locations[2].Set(42.01f, -2.13f, 0f);//row 2
locations[4].Set(42.01f, -8.84f, 0f);//row 3

this.transform.position = locations[3];
```

```
4 references
public void setText()
{
    //sets the scores and cookies into the game space
    scorecountText.text = "Score: " + score.ToString();
    sCookieCountText.text = "Small Cookies: " + smallCookies.ToString();
    bCookieCountText.text = "Big Cookies: " + bigCookies.ToString();

}
```

```
if (this.transform.position == locations[2] || this.transform.position == locations[0] || this.transform.position == locations[4])
{
    //this throws the small cookie when E or V are pressed
    if (Input.GetKeyDown(KeyCode.E) || Input.GetKeyDown(KeyCode.V))
    {
        if (gameManager.smallCookies > 0)
        {
            Rigidbody2D clone;
            //clones a small cookie box, and sends it to the left at a certain speed
            clone = Instantiate(Smallbox, transform.position, transform.rotation);
            clone.AddForce(CookieSpeed);
            gameManager.smallCookies--;

        }
    }
    //if Q is pressed, player throws a big cookie and reduces big cookie count
    if (Input.GetKeyDown(KeyCode.Q) || Input.GetKeyDown(KeyCode.C))
    {
        if (gameManager.bigCookies > 0)
        {
            //clones a big cookie box then sends it to the left.

            Rigidbody2D clone2;
            clone2 = Instantiate(Bigbox, transform.position, transform.rotation);
            clone2.AddForce(CookieSpeed);
            gameManager.bigCookies--;
        }
    }
}
```