GAME DESIGN DOCUMENT

By Piyush Pallav & Akshay Bharadwaj

CONCEPT BRIEF

PLOT: Lets take care of a very hungry baby. Only when you feed the baby the right things, he/she will stop crying.

BRIEF: In this game, you control the spoon and collect food items while avoiding obstacles/harmful products to feed the hungry baby at the end of the track.

GENRE: Hyper casual game | **SUB GENRE:** Arcade (19% popularity in Feb 2023)

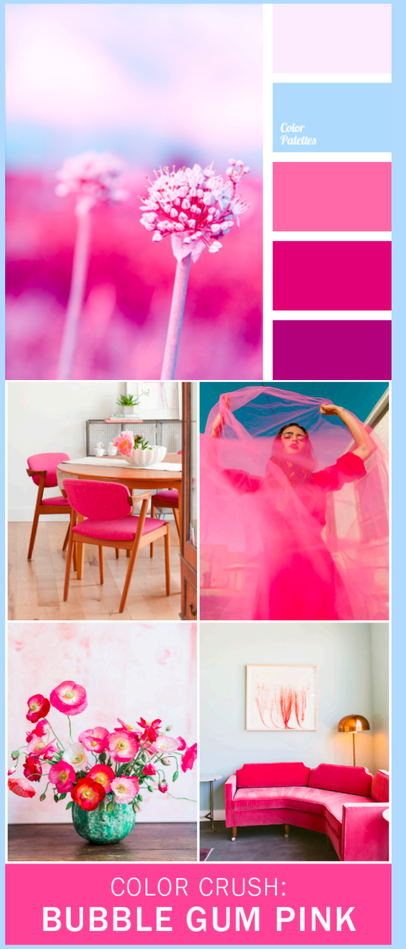
THEME: Food, balls, Toys, Kids

MECHANICS: LEFT and RIGHT flying mechanics; Scoring and collection mechanics; Physics mechanics for obstacles and the player character itself.

SCORING: Every collectible adds a point and every obstacle subtracts a points In the end depending on the score, the player gets a Star rating.

CONTROLS: LEFT and RIGHT movement using dragging controls (Most popular in Feb 2023).

ORIENTATION: Portrait view where camera is behind the character.



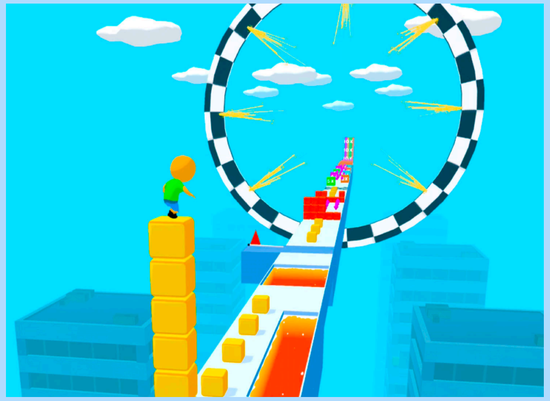
CONCEPT BRIEF

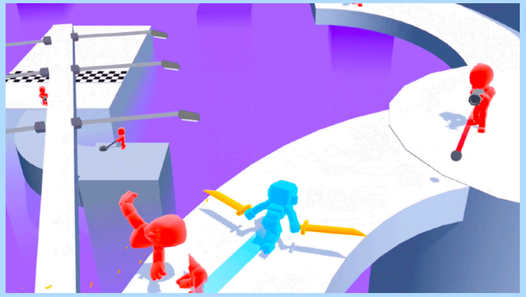
REFERENCE TREND: OPEN LINK

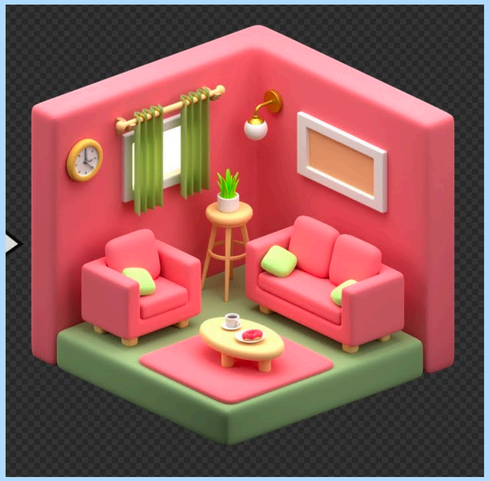
WOW MOMENTS: A sense of flying through various beautiful 3D levels; Effects on point collection, collision with obstacles and reaching the end.

LEVELS: Levels will be different rooms in the house like the bathroom, the baby's bedroom, kitchen, dining room, etc.

FUTURE FOCUS: Every room will have 5 stages with different layouts and increasing difficulty; New unlock-able outfits for the baby through in game purchase.

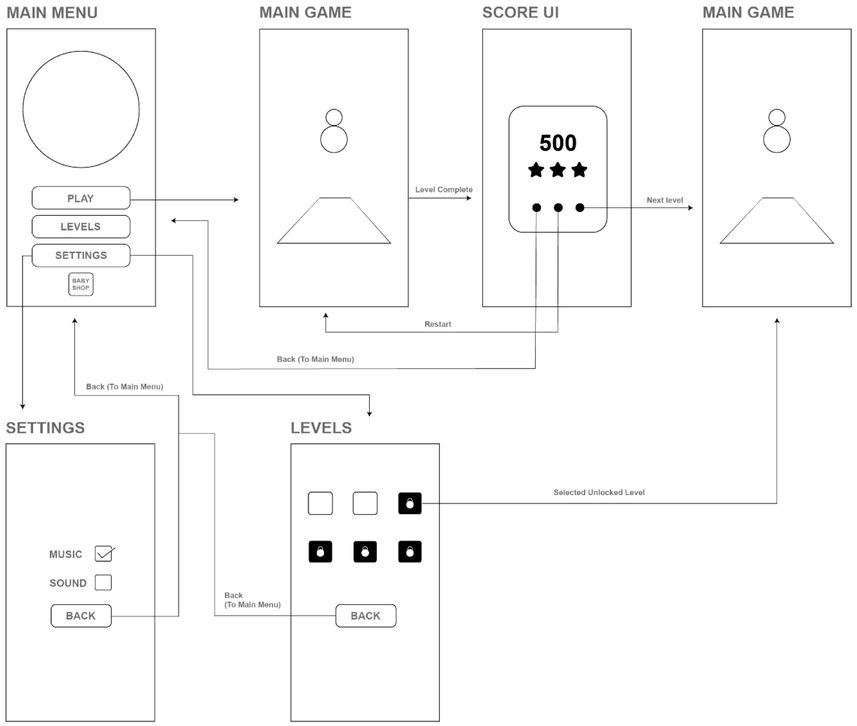






CONCEPT BRIEF

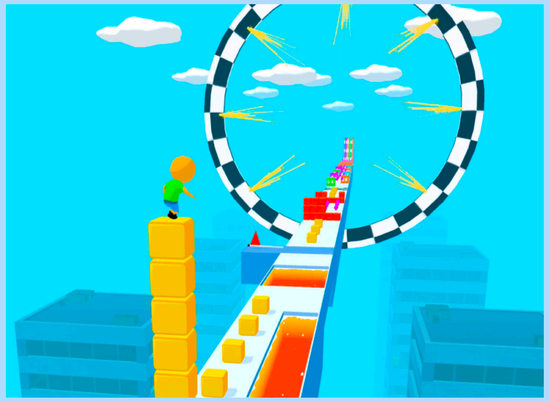
WIREFRAME: The basic wireframe of the game.

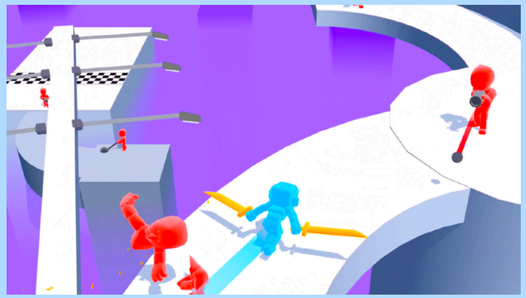


ASSET PREPARATION

ART BIBLE

Art style : The game aims for a vibrant and playful visual using simple 3D elements using basic 3D shapes that are very cute and comfy.





Font Family: The font which we chosen is **"Bubblebody"**. "The menus, UI, popups will use the same font. This font gives goes with the comfy and cute vibe of the game.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

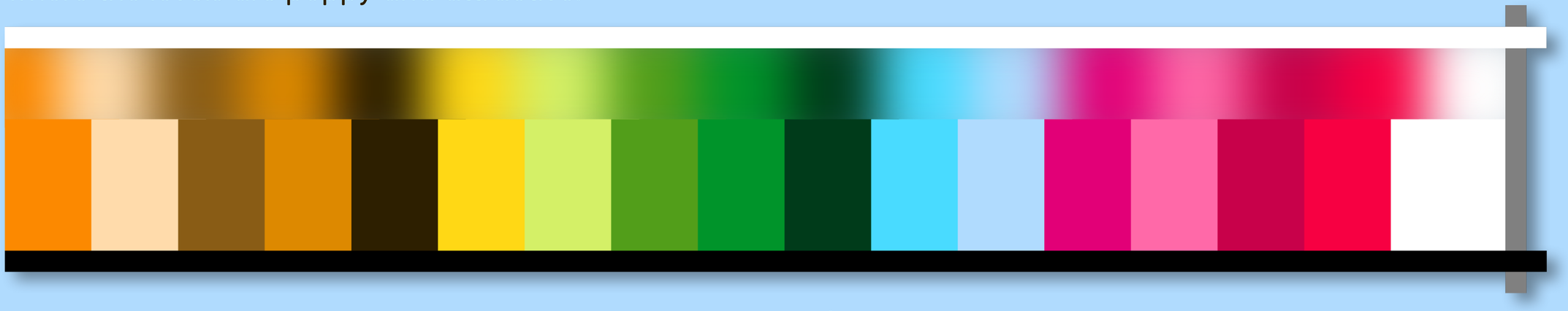
a b c d e f g h i j k l m n o p q r s t u v w x y z

1 2 3 4 5 6 7 8 9 0

ASSET PREPARATION

ART BIBLE

Colour Schemes: Our Eye catchy colour scheme is again using saturated colours where the levels are poppy and attractive.



Character Art: The baby is the cutest main character. He/she is made using 2D illustrations made on Illustrator which is then wrapped on 3D models (made on Maya).

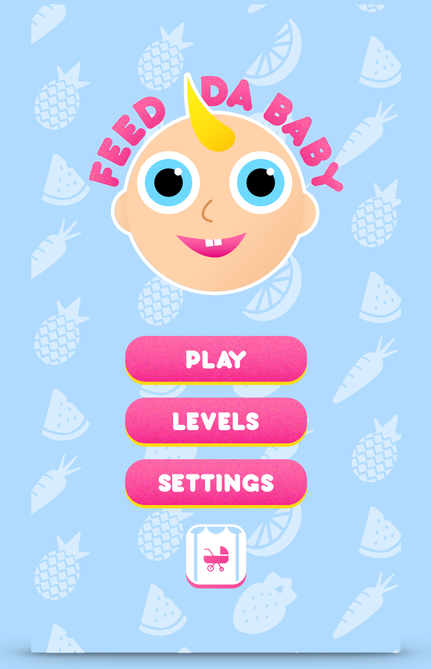
ASSET PREPARATION

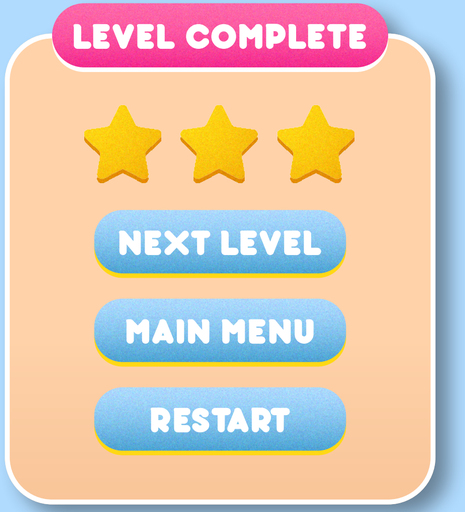
ART BIBLE

Environments and Camera View: The first person camera view is like holding something in front of your phones back camera and recording it. The environments are made of a background wall and floor and props to give a sense a depth to the player.

ASSET PREPARATION

UI & GUI: The UI is made of buttons and banners on panels for pops, and menu screens. Curved edges and vibrant colours give a very playful feel to the UI.

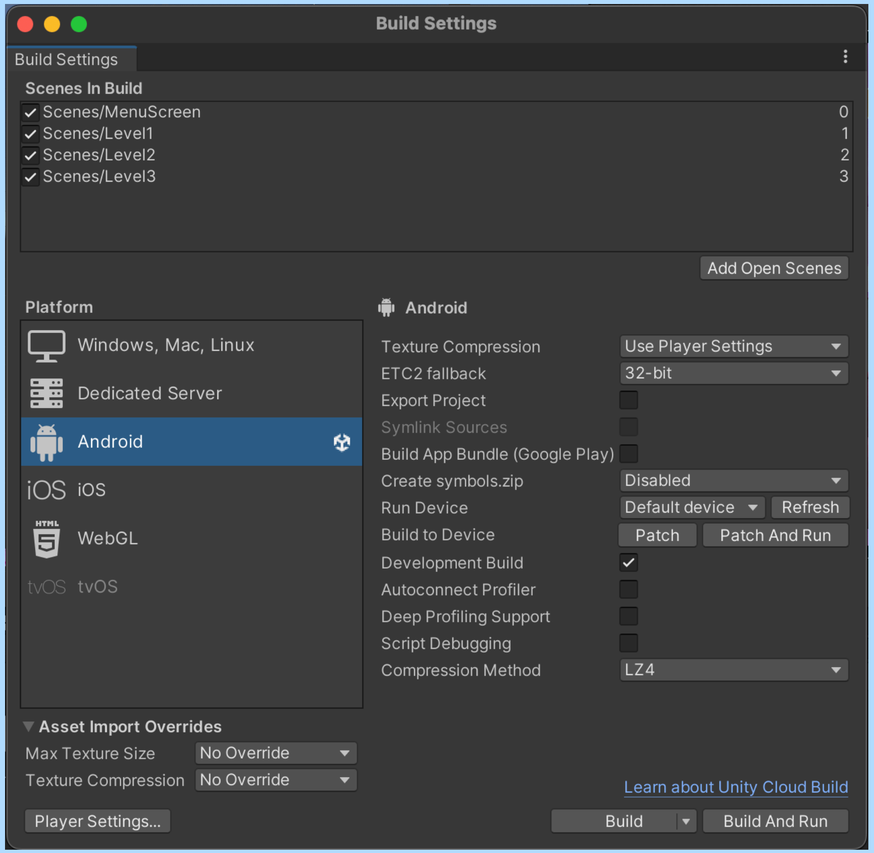




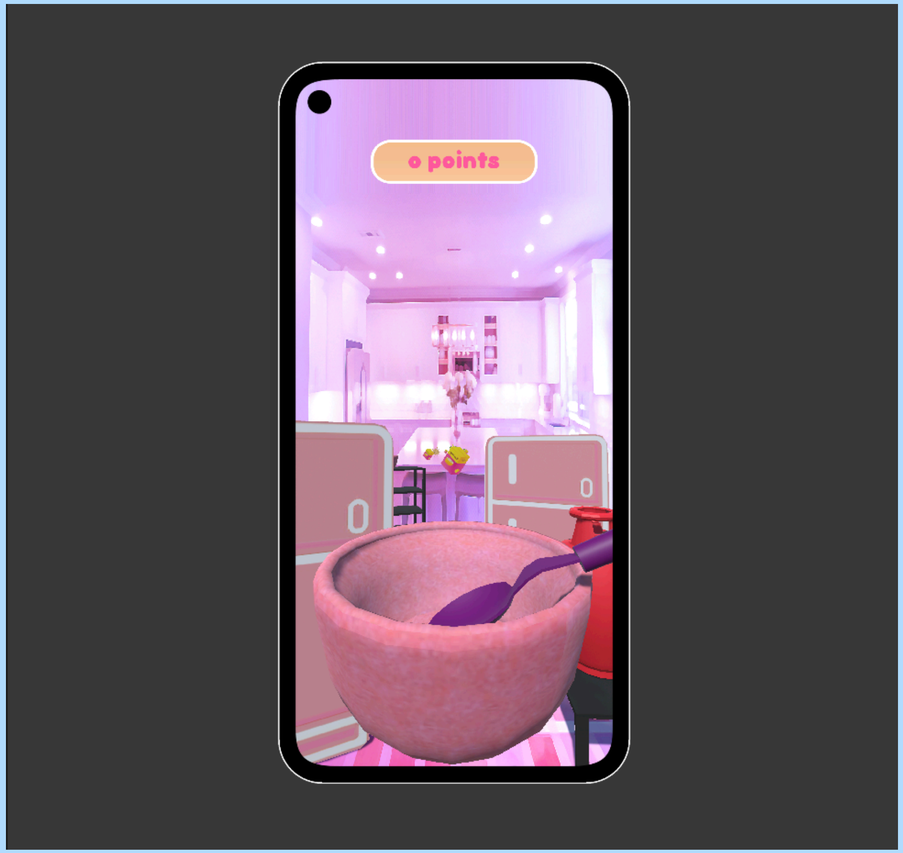
SOUND EFFECTS: There are 3 basic sound effects. Background music, Obstacle/ collectibles SFX, UI SFX.

INTEGRATION

BUILD SETTINGS: Basic Android Build Settings



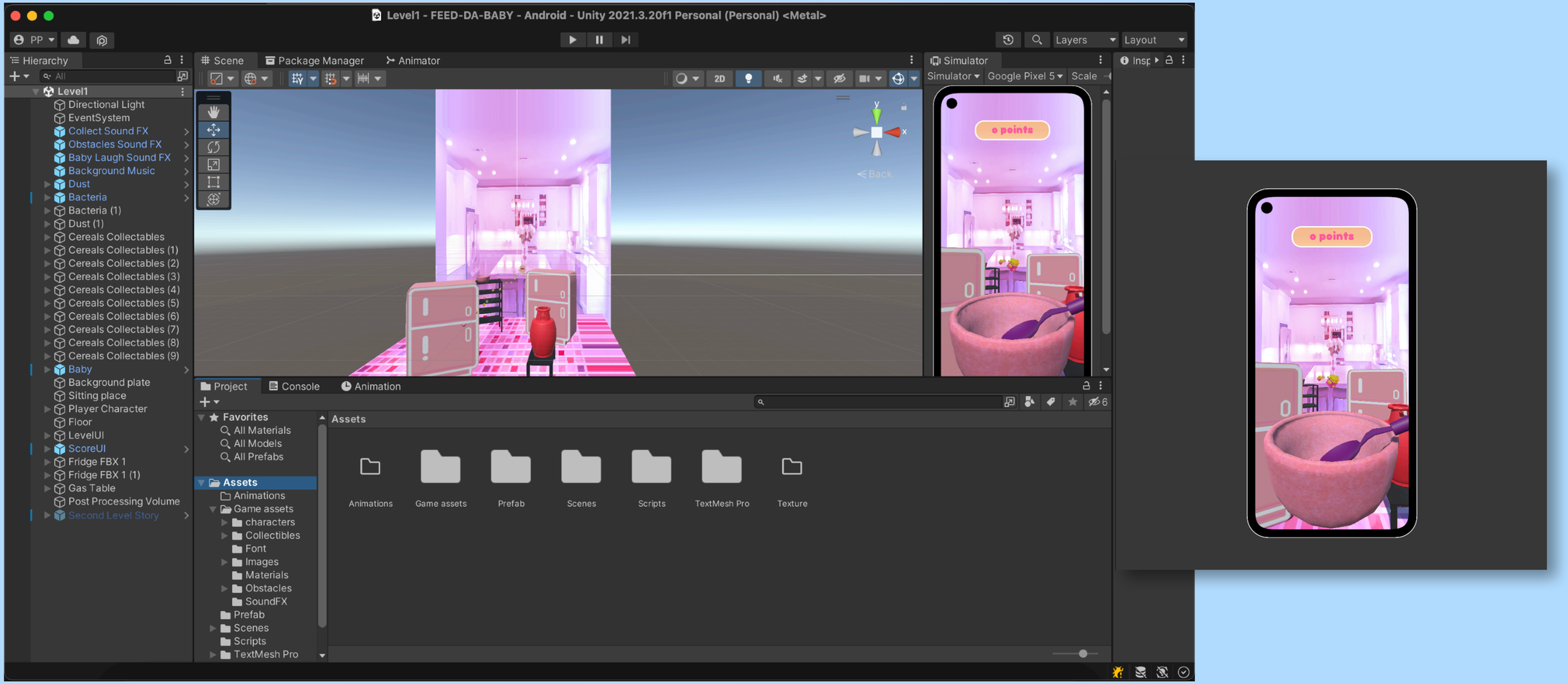
ORIENTATION: Portrait.



INTEGRATION

ASSET INTEGRATION: Added assets to the project.

CAMERA VIEW: First person.



INTEGRATION

PHYSICS & INPUT CONTROLS


```
public class PlayerMovement : MonoBehaviour
{
    [SerializeField]
    private float speed = 0.01f;
    private Touch touch;
    [SerializeField]
    private float movementSpeed;
    private float minX;
    private float maxX;



    // Update is called once per frame
    void Update()
    {

        if (Input.touchCount > 0)
        {
            touch = Input.GetTouch(0);

            if(touch.phase == TouchPhase.Moved)
            {
                transform.position = new Vector3(transform.position.x + touch.deltaPosition.x * speed,
                transform.position.y, transform.position.z);

                float newX = transform.position.x + touch.deltaPosition.x * speed;

                float clampedX = Mathf.Clamp(newX, -3.5f, 5.4f);
                transform.position = new Vector3(clampedX, transform.position.y, transform.position.z);
            }


        }
        transform.Translate(0, 0, 1 * movementSpeed * Time.deltaTime);

    }
}
```

INTEGRATION

PHYSICS & INPUT CONTROLS

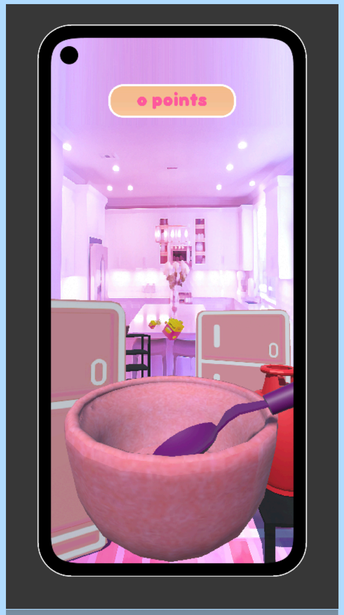
```
private void OnTriggerEnter(Collider collision)
{
    if (collision.gameObject.CompareTag("Items"))
    {
        collectiblesSfx.Play();
        Destroy(collision.gameObject);
        score++;
        scoreText.text = score.ToString() + " " + "points";

        if (score >= 1 && score <= 6)
        {
            starOne.SetActive(false);
            starTwo.SetActive(true);
            starThree.SetActive(false);
        }

        if (score > 6 && score <= 12)
        {
            starOne.SetActive(true);
            starTwo.SetActive(true);
            starThree.SetActive(false);
        }

        if (score > 12 && score <= 20)
        {
            starOne.SetActive(true);
            starTwo.SetActive(true);
            starThree.SetActive(true);
        }
    }

    if (collision.gameObject.CompareTag("Obstacle"))
    {
        obstaclesSfx.Play();
        Destroy(collision.gameObject);
        score--;
        scoreText.text = score.ToString() + " " + "points";
    }
```

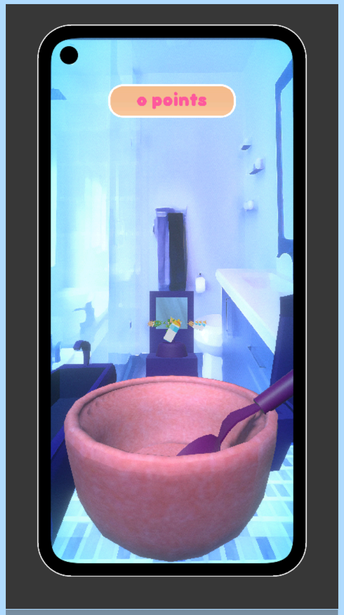
```
private void OnTriggerEnter(Collider collision)
{
    if (collision.gameObject.CompareTag("Items"))
    {
        collectiblesSfx.Play();
        Destroy(collision.gameObject);
        score++;
        scoreText.text = score.ToString() + " " + "points";

        if (score >= 1 && score <= 3)
        {
            starOne.SetActive(false);
            starTwo.SetActive(true);
            starThree.SetActive(false);
        }

        if (score > 3 && score <= 7)
        {
            starOne.SetActive(true);
            starTwo.SetActive(true);
            starThree.SetActive(false);
        }

        if (score > 7 && score <= 10)
        {
            starOne.SetActive(true);
            starTwo.SetActive(true);
            starThree.SetActive(true);
        }
    }

    if (collision.gameObject.CompareTag("Obstacle"))
    {
        obstaclesSfx.Play();
        Destroy(collision.gameObject);
        score--;
        scoreText.text = score.ToString() + " " + "points";
    }

    if (collision.gameObject.CompareTag("Child"))
    {
        babyGiggle.Play();
        Destroy(collision.gameObject);
        backgroundMusicfx.Stop();
    }
}
```

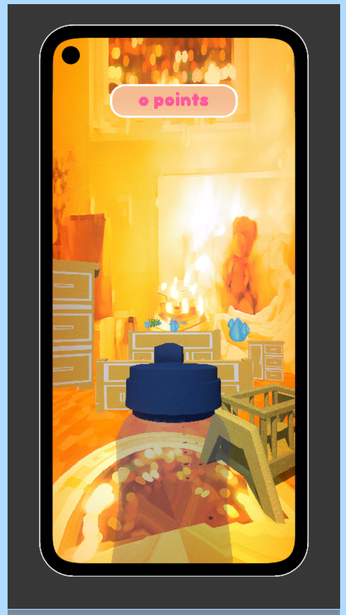
PROTOTYPE

To check the prototype gameplay, CLICK HERE!

LEVELS: There are total of 6 levels(3 still in Dev. Phase). Each







Dev.ph.

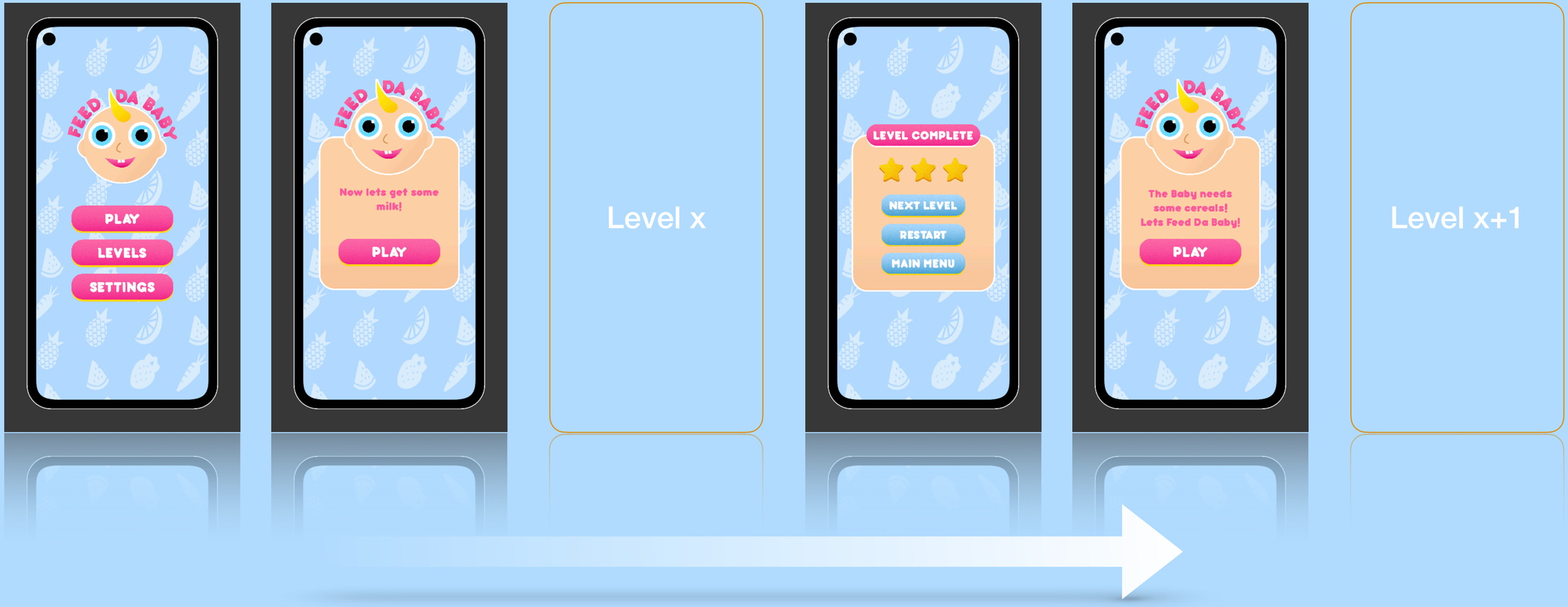
Dev.ph.

Dev.ph.

PROTOTYPE

To check the prototype gameplay, CLICK HERE!

UI: The levels will be preceded and followed by GUI screens which will guide the player through the story and the game play of the game.

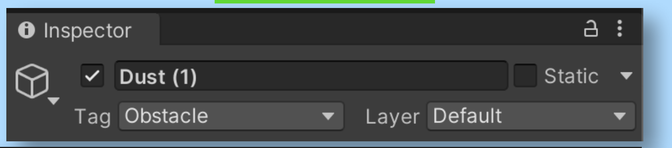


PROTOTYPE

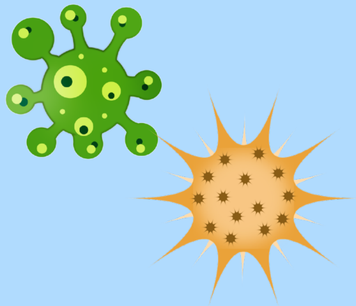
To check the prototype gameplay, CLICK HERE!

SCORING: The collectibles have been give tags in unity of “Items” and obstacles have been given “Obstacles”. Fetching their tags through scripts, score is either added or subtracted finally giving a star rating.

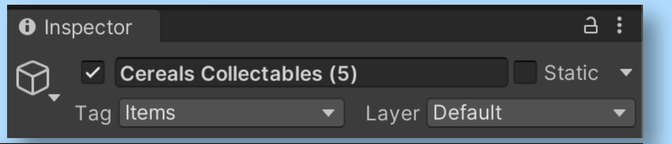
+1




```
if (collision.gameObject.CompareTag("Items"))
{
    collectiblesSfx.Play();
    Destroy(collision.gameObject);
    score++;
    scoreText.text = score.ToString() + " " + "points";
```



-1




```
if (collision.gameObject.CompareTag("Obstacle"))
{
    obstaclesSfx.Play();
    Destroy(collision.gameObject);
    score--;
    scoreText.text = score.ToString() + " " + "points";
}
```

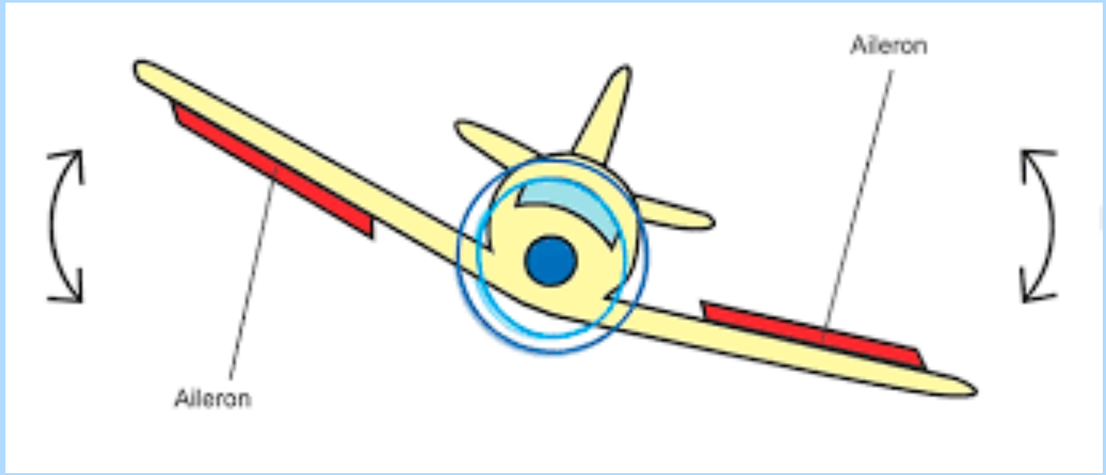
PROTOTYPE

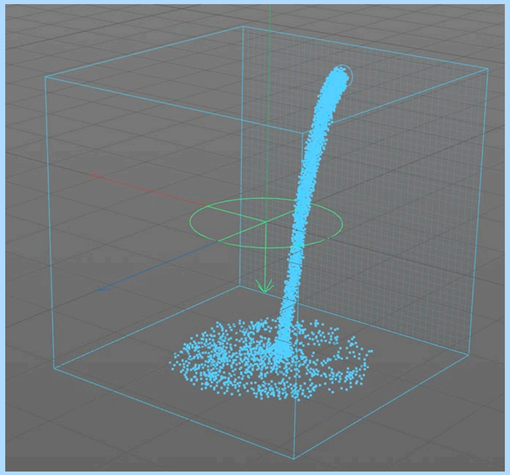
To check the prototype gameplay, CLICK HERE!

UPDATED MECHANICS: There are 2 mechanics which are essential to the game and are being added :

-**Container filling with food/water/medicine/juice using particle mesh renderer,**
Issue to be resolved: Figuring out the workflow of the particle rendering.

-and rolling the container on its axes to give it a feeling of flying.
Issue to be resolved: When dragged right/left, the player character/container starts spinning and just goes diagonally off the straight track.

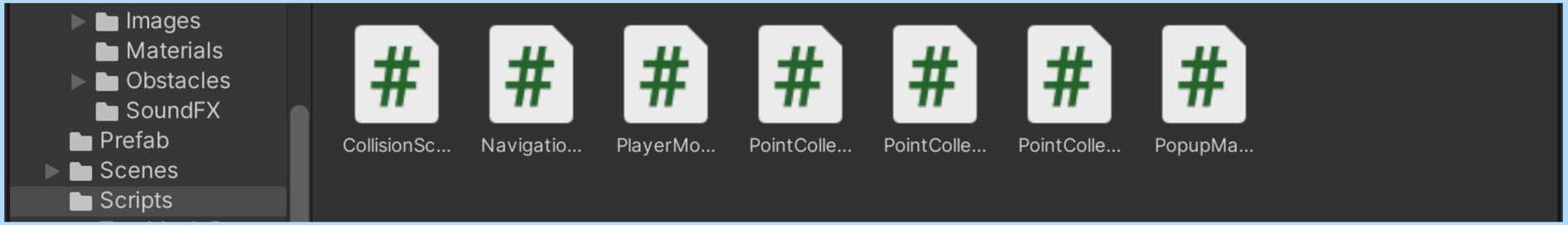




POLISHING AND OPTIMISATION

To check the prototype gameplay, CLICK HERE!

CODE DEBUGGING AND OPTIMISATION: Throughout the process the various scripts and their rolls have been improved. As of now there are 5 kind of scripts: Player Movement, Collision, Point collection, Level UI, GUI navigation. To increase difficulty in levels, we are going to add Obstacle/Collectible Movement script(s) as well.



We use Github to commit our versions of our work to make sure nothing is lost.

UPDATED TRACKER: As a small team, the two of us divide our work and set deadlines for each other to push each other to get the work done asap. TRACKER LINK!

LEGEND>>>		> Not Started!
		> WIP
		> Completed!
	DD/MM/YY	> Late Delivery
	DD/MM/YY	> OT Delivery
	DD/MM/YY	> BT Delivery