(Week 10)

การเขียนโปรแกรมคอมพิวเตอร์ขั้นสูงเพื่อควบคุมอุปกรณ์

ADVANCE COMPUTER PROGRAMMING

สอนโดย พงศธร เกียรติเจริญพร (มิว)


User experience
Design

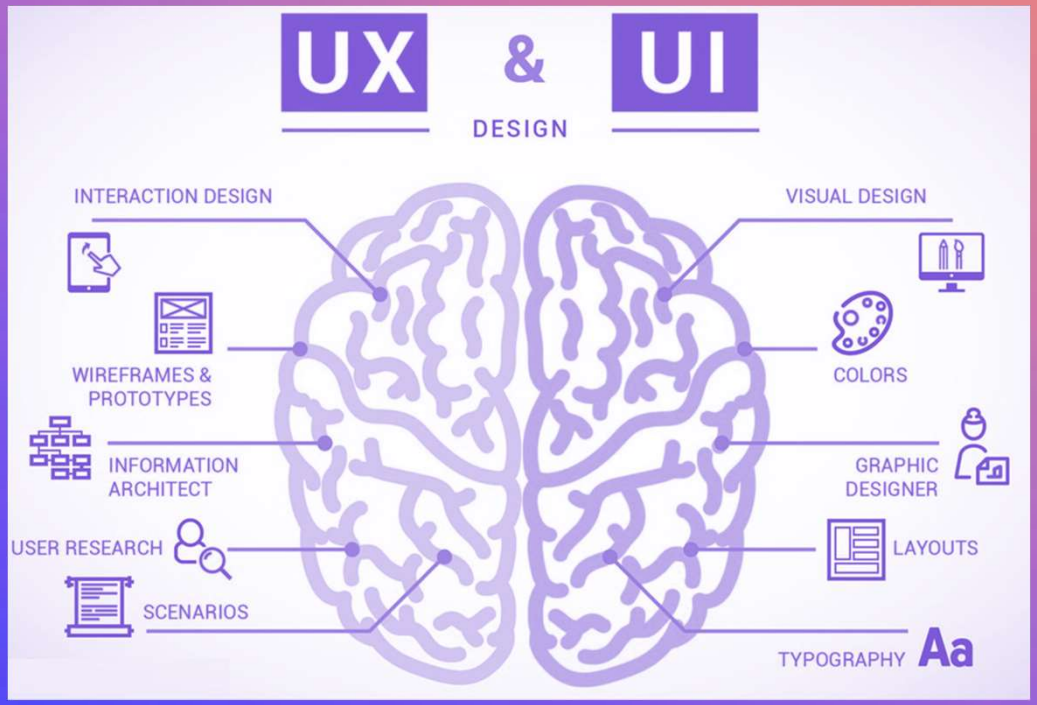
UX ไม่ใช่ UI

UX: User Experience Design

เป็นศาสตร์ที่ว่าด้วยประสบการณ์ใช้งานของผู้ใช้ ซึ่งจะเป็นเรื่องของการแก้ปัญหาในการใช้งาน ความรู้สึกของผู้ใช้ ให้ใช้งานได้สะดวกและตอบโจทย์ความต้องการของผู้ใช้มากยิ่งขึ้น

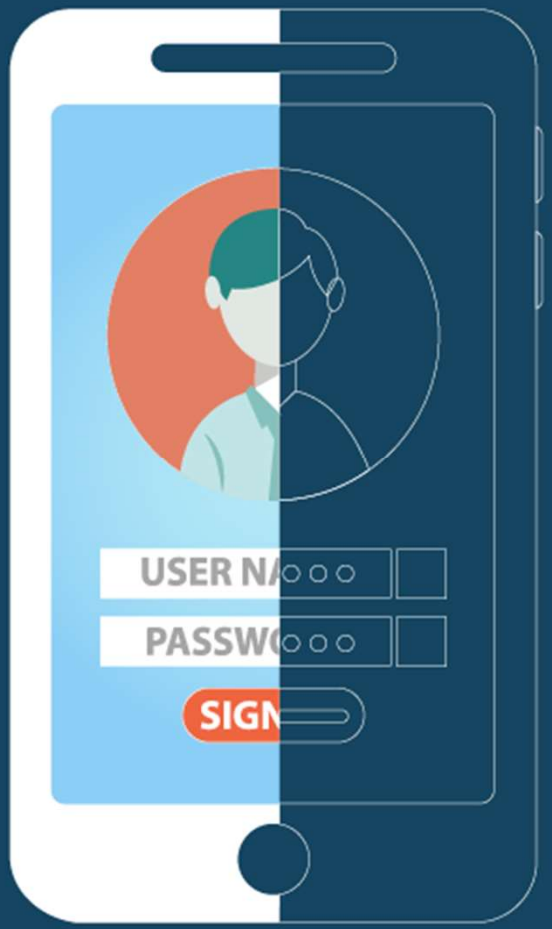
UI หรือ User Interface Design

นั้นก็คือศาสตร์ที่มาเติมเต็มให้กับ UX นั้นสวยงามและสมบูรณ์ ไม่ว่าจะเป็นเรื่องการจัดวางองค์ประกอบต่างๆ สี ขนาดตัวอักษร ฟอร์มต่างๆ ที่จะไปอยู่ภายในเว็บไซต์ หรือแอพพลิเคชั่นของเราให้สะอาด สวยงาม



UX [illegible] สิ่งที่เน้นความสวยงาม

UI



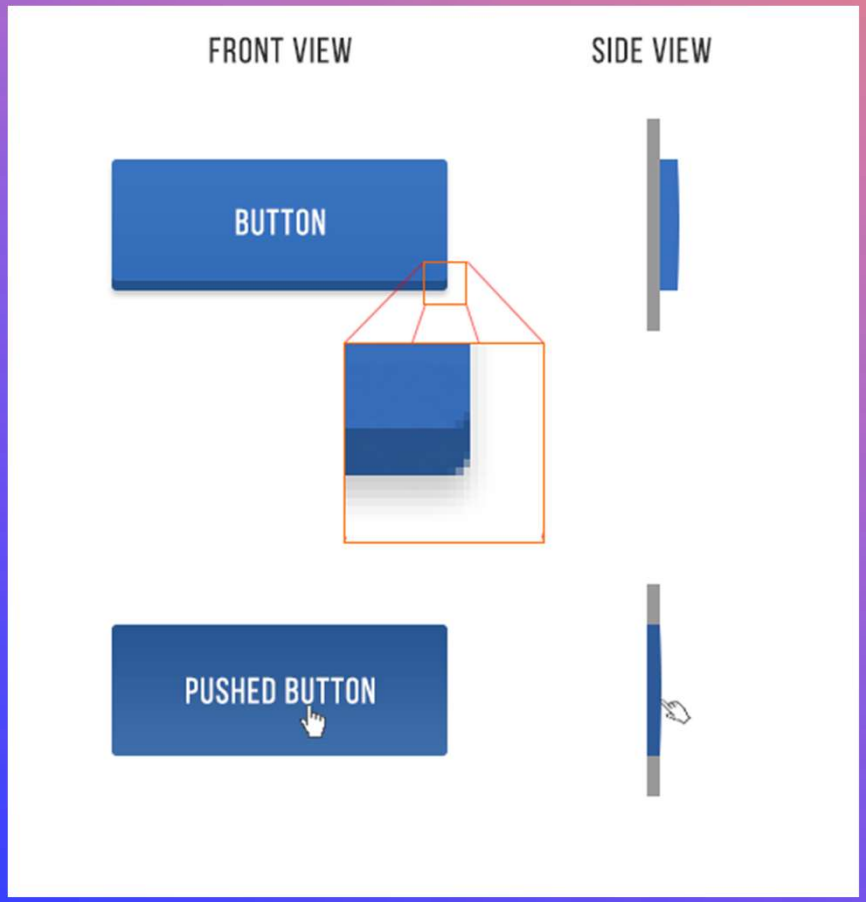
UX

User Interface

การออกแบบ interface เป็นสิ่งหนึ่งที่ท้าทายมากที่สุดในการพัฒนาเกม เพราะว่ามันมีข้อมูลเป็นจำนวนมากที่ต้องถ่ายถอดให้ผู้เล่น บนจอที่มีพื้นที่ไม่มาก

7 กฎการออกแบบ UI

1. แสงต้องมาจากบนฟ้า
2. เริ่มด้วย ขาว-ดำ เสมอ
3. เพิ่ม Whitespace (ช่องว่าง) เป็นสองเท่า
4. เทคนิคการวางตัวหนังสือบนรูป ไม่ให้จม
5. เพิ่ม – ลด ความเด่นของตัวหนังสือ
6. เลือกใช้ฟ้อนต์ให้เหมาะสม
7. ขโมยอย่างศิลปิน



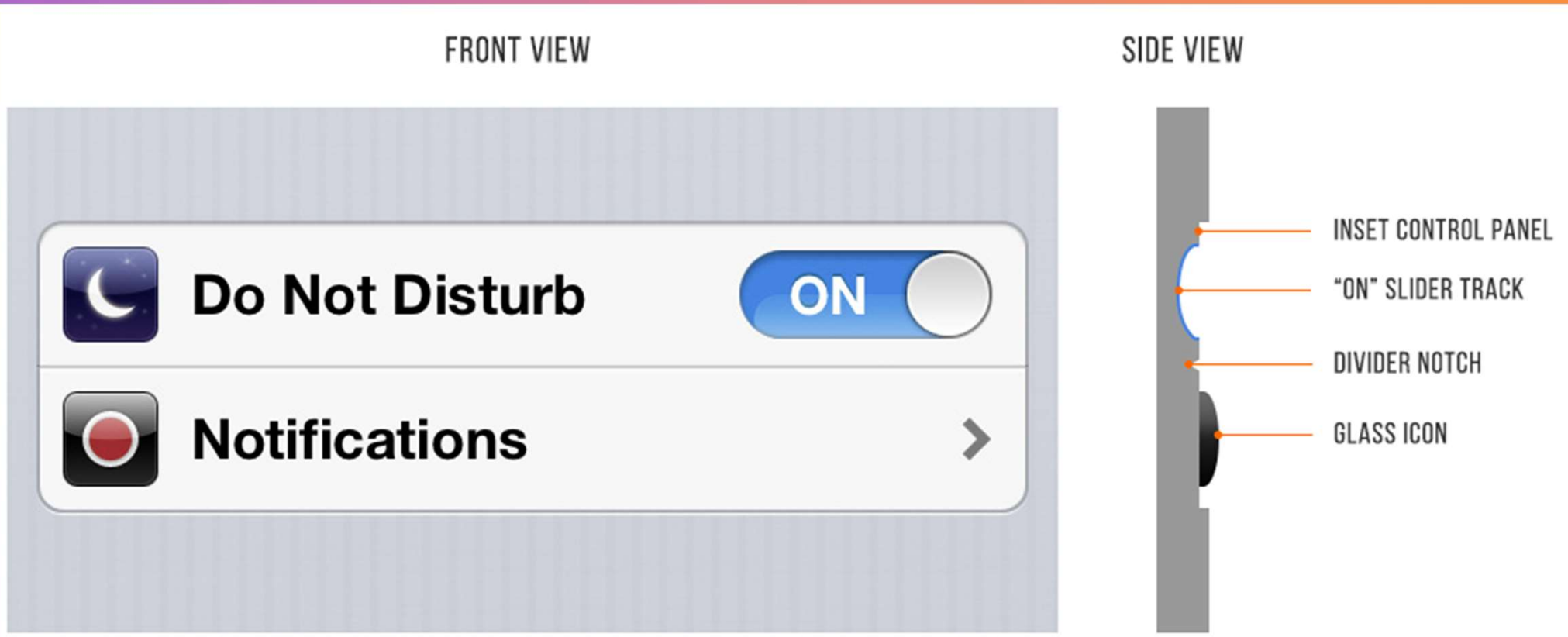
ด้านบนของสิ่งที่โดนแสงจะสว่าง และ
ด้านล่างจะมืดกว่าเสมอ

การออกแบบ User Interface ก็เป็นไป
ตามเรื่องแสงนี้เช่นกัน สังเกตในจะเห็นว่า
UI หลาย ๆ จุด เช่น ปุ่ม จะมีการใส่ “เงา”
เพื่อให้เรารู้สึกว่าปุ่มเป็น 3 มิตินูนออกมา

ถึงแม้มันจะดูเรียบ ๆ สไตล์ Flat Design แต่ก็มีการใช้
รายละเอียดเรื่องแสงเข้ามาเกี่ยวข้อง

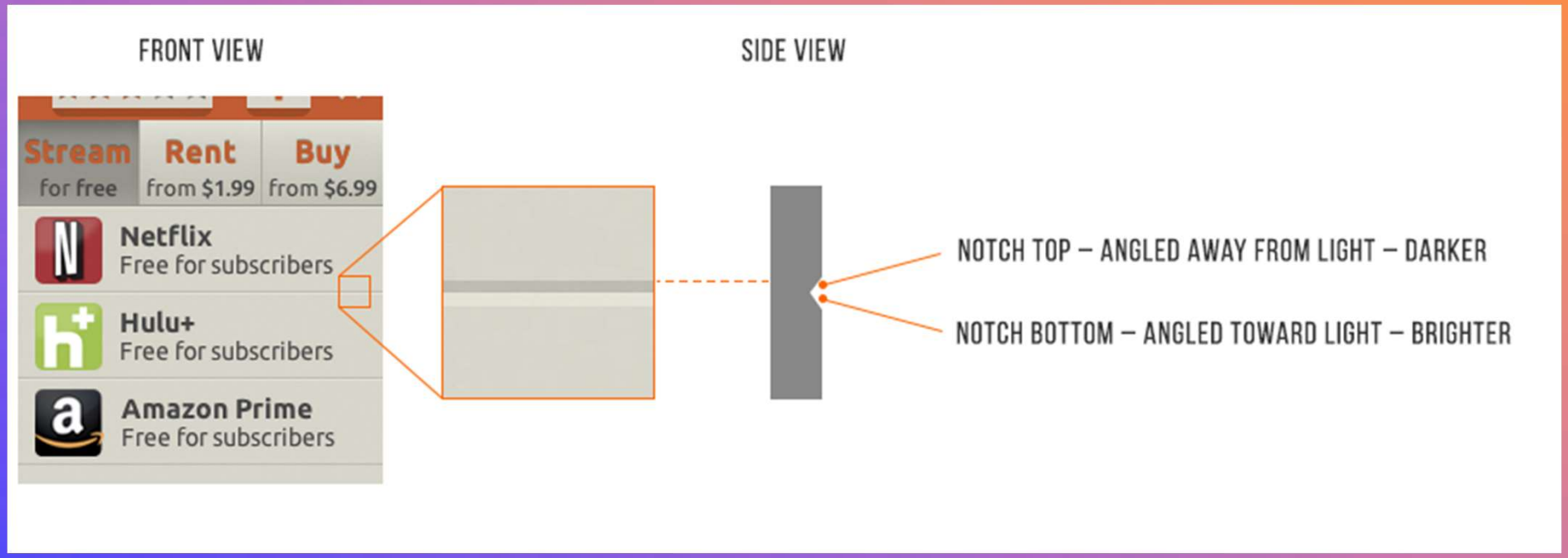

VICTORY!
Level 12: Revange
Turns to Win:
122
Total Score:
3
Achievements:
Next Level
DEFEAT!
Level 1: Revange
Mind using Power-Ups:
Go to Map
Try Again


Info
Blood Pacts
The Great Truce
Blood Pact signed
Gorge of Whispers
Level 0/20
Blackrock
Level 1/20
Plain of Blades
Level 1/5
Sappers' Guild - I
Level 0/1
Whitefall
Level 1/20
Highcliff
Level 1/20
@小金狮-UI动物学园


FRONT VIEW
SIDE VIEW
Do Not Disturb
ON
Notifications
INSET CONTROL PANEL
"ON" SLIDER TRACK
DIVIDER NOTCH
GLASS ICON


HERO
Attribute list
Jack Sparrow
Note that this is a very dangerous role!
HP
100/200
MP
150/200
Exp
130/200
Lv.12
Lv.12
Lv.10
Lv.8
Jack Sparrow
QQ 370328021
报名群：66840852
@deviljack-99（网络班）


options
3500
315
QUIT

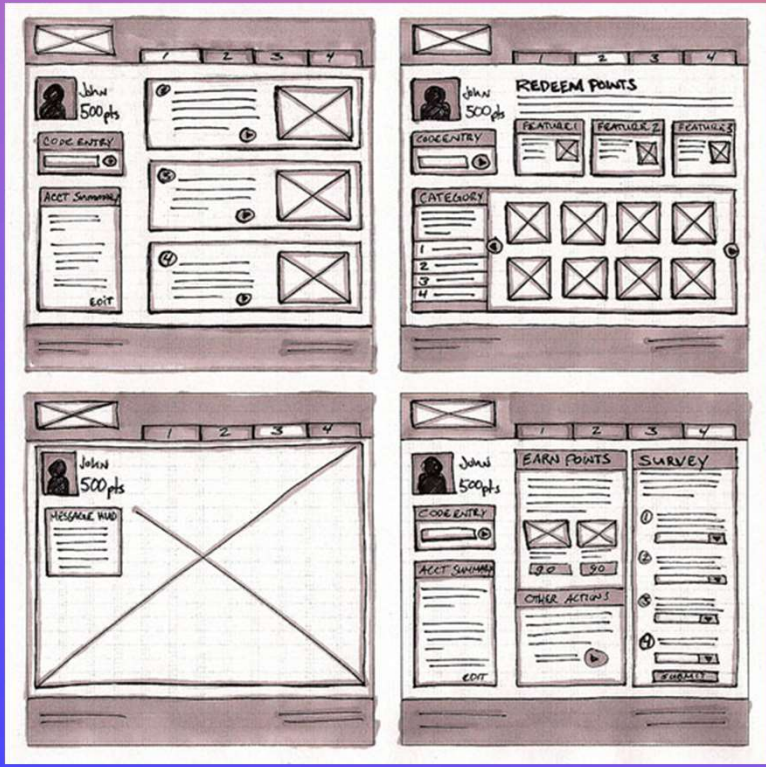

FRONT VIEW
SIDE VIEW
Stream
for free
Rent
from $1.99
Buy
from $6.99
Netflix
Free for subscribers
Hulu+
Free for subscribers
Amazon Prime
Free for subscribers
NOTCH TOP – ANGLED AWAY FROM LIGHT – DARKER
NOTCH BOTTOM – ANGLED TOWARD LIGHT – BRIGHTER

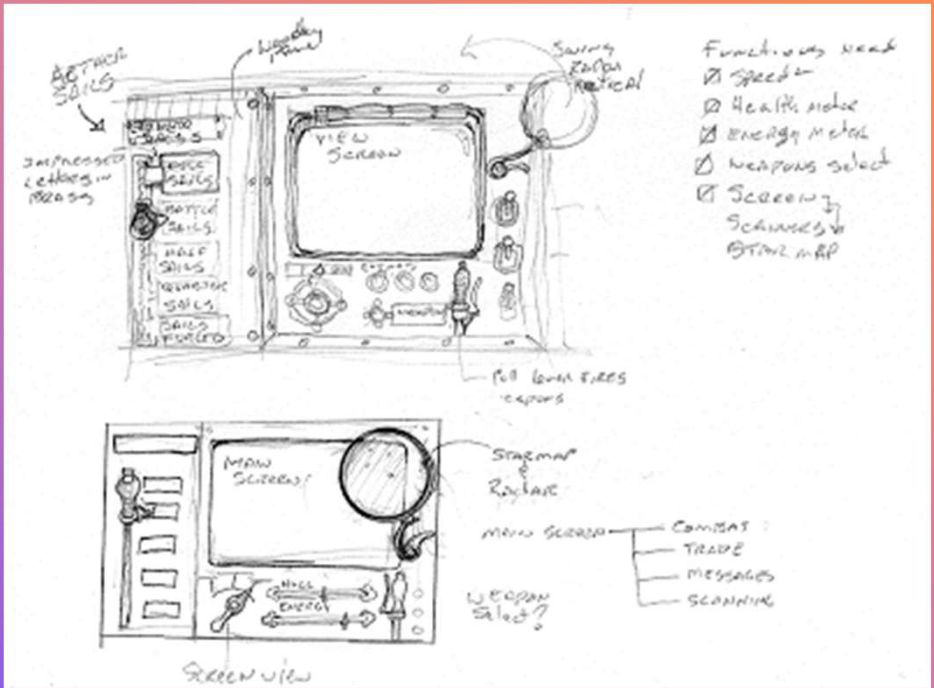

HALO
WAYPOINT
TEAM RESULTS
1ST
2ND
3RD
4TH
5TH
6TH
7TH
8TH
TEAM COMPARISON
ITZZ TOURIST
SR 22
SPARTAN-IV


STAGE 12
HUNGRY APP
999.9 B
999.9 B
999.9 B
LV. 4
초보물약
+300
310A
5%
사용
바나나처럼 휜 칼
Lv22
체력 강화
Lv.06
방패 강화
Lv.00

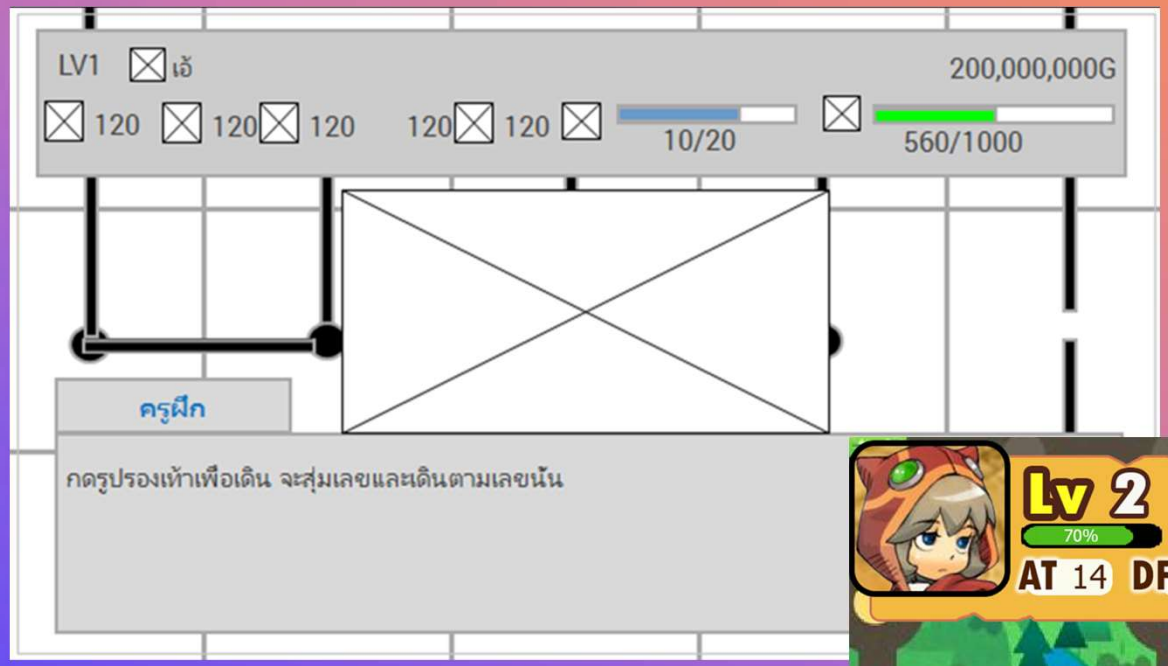
Material Design โดยเน้นการแสดงความลึกของสิ่งต่าง ๆ ด้วยแสงเงา เลียนแบบเงาของจริง

ดีไซน์ UI ที่ซับซ้อนออกมาได้ง่ายขึ้น และนอกจากนั้นยังทำให้เราโฟกัสกับเรื่องการจัดช่องวาง และวาง Layout สิ่งต่าง ๆ ก่อนด้วย



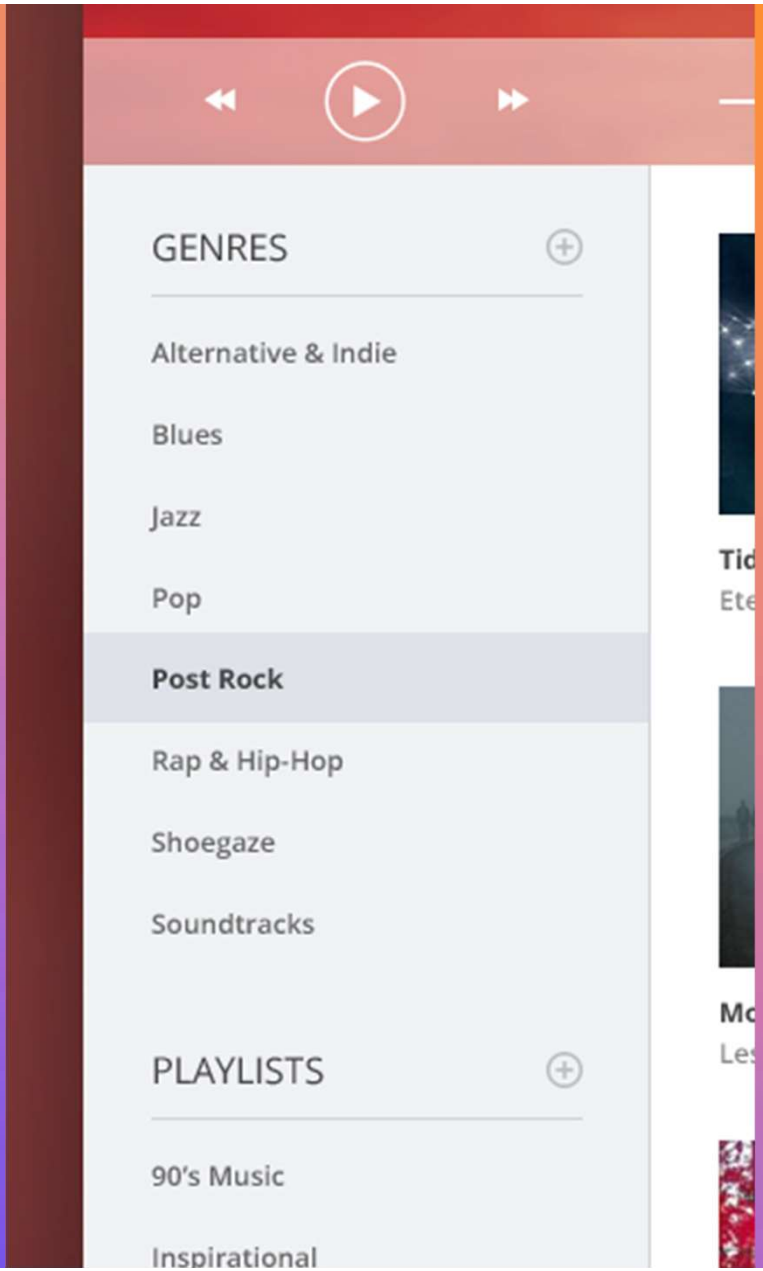


การใส่สีทุกครั้งต้องมีจุดประสงค์ในการใส่


LV1 เอ้
200,000,000G
120 120 120 120 120
10/20
560/1000
ครูฝึก
กดรูปรองเท้าเพื่อเดิน จะสุ่มเลขและเดินตามเลขนั้น


Lv 2
70%
Aliza
35
10,400G
HP 42 / 50
SP 25 / 50
04:24 ฟื้นฟูทั้งหมด
AT 14 DF 20 MG 5 DF 25
5
6 1
XXXXXXXXXXXXXXX XXXXXXX XX??
XXXXXXX XXXXXXX XXXXXX
XXXXX

เพิ่ม WHITESPACE เป็น 2 เท่า




PROFILE
BROTHERHOOD OF MOON
Lv. 12 Nemesis450
CHARACTERISTICS
Empower 100%
Magic defense 100%
Concentration 100%
Evasion 100%
Max HP 220
Critical damage 20%
Critical chance 20%
Total XP: 12345678
Achievments: 45/100


UIZOO.CN
66666
66666
Hero Raiders
Bryant Ust
Viability
Aggressivity
Difficulty
Story
Equipment
Poster
Attribute
Skill
Skin
PROPHET Wade
Wade Classic
Already dressed
Need to get the hero
UIZOO.CN
UI 实体班
小金狮
Des:奥特漫

หลักๆ:

1. รูปพื้นหลังต้องสีเข้ม – และมีจุดที่สีตัดกันไม่เยอะ (สมมติถ้าพื้นหลังเป็นตารางหมากรุกขาว – ดำจะวางยาก)
2. ตัวหนังสือต้องสีขาว – ถ้าอยากได้ลุคสวย ๆ คลีน ๆ จะเป็นตัวหนังสือสีขาวสว่าง ตัวหนังสือดำก็ได้ แต่ข้อ 1. จะตรงข้ามกัน คือ รูปพื้นหลังต้องสว่างมาก
3. ทดสอบทุกขนาดหน้าจอ – ว่าย่อขยายแล้วมีขนาดหน้าจอไหนที่ตัวหนังสืออ่านไม่ออกมั้ย


Smithy
157/95
2,299,937
297
Weapon Refine
Armor Refine
Accessory Refine


MOVE
MISSION
SCORE
38
12
8,888,888
8
30
3

22/09/2021

เพิ่ม – ลด ความเด่นของตัวหนังสือ

- เปลี่ยนขนาดตัวหนังสือ
- เปลี่ยนสีตัวหนังสือ
- ปรับตัวหนา-บาง
- ใช้ตัวพิมพ์เล็ก – ตัวพิมพ์ใหญ่
- ใช้ตัวเอียง
- เปลี่ยนระยะห่างตัวหนังสือ
- เปลี่ยนระยะห่างของกล่อง (ทำให้เกิด Whitespace ที่ทำให้ตัวหนังสือดูดีได้)
- การใช้ตัวหนังสือมีหาง (Serif) – ไม่มีหาง (Sans-serif)


TIPS
MONSTERS
TOWERS
Monsters
Flying Monsters
Fatty
Tin Boss
SPEED
SPEED
SPEED
SPEED
HP
HP
HP
MAX
HP
※ JUST AN EXAMPLE OF A LARGE MONSTER LEGION

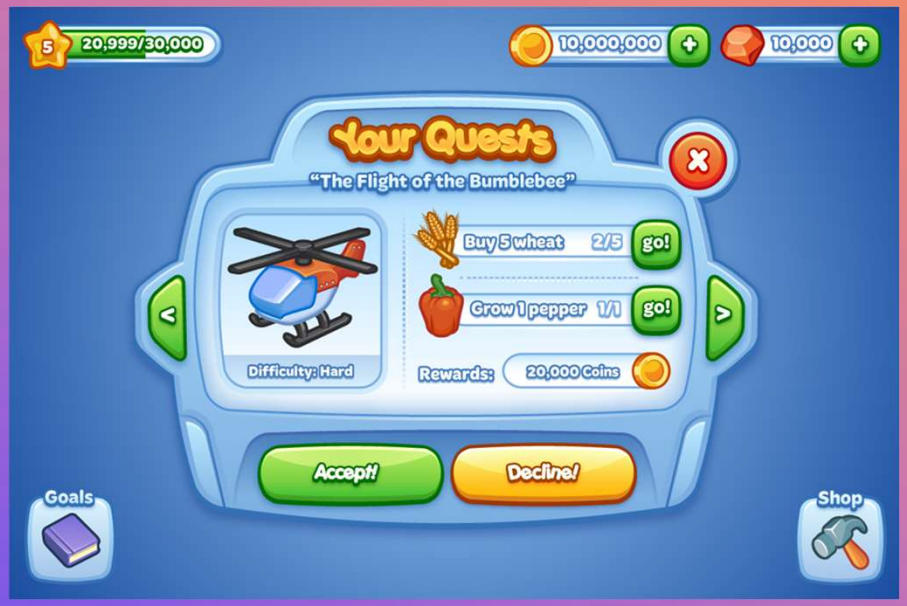

STAGE PAGE
CHAPTER 56
1,240
65/100
12,123,450
OAK WORRIOR
STAGE 27 hard
Dancing Oak Worrior
Once upon a time, the communication media sometimes called the mess media
BOSS
Lv.89 Oak Worrior
RECOMMENDED POWER
91,850 (Titration)
APPEAR MOSTERS
Compensation
EXP
PLAY

22/09/2021

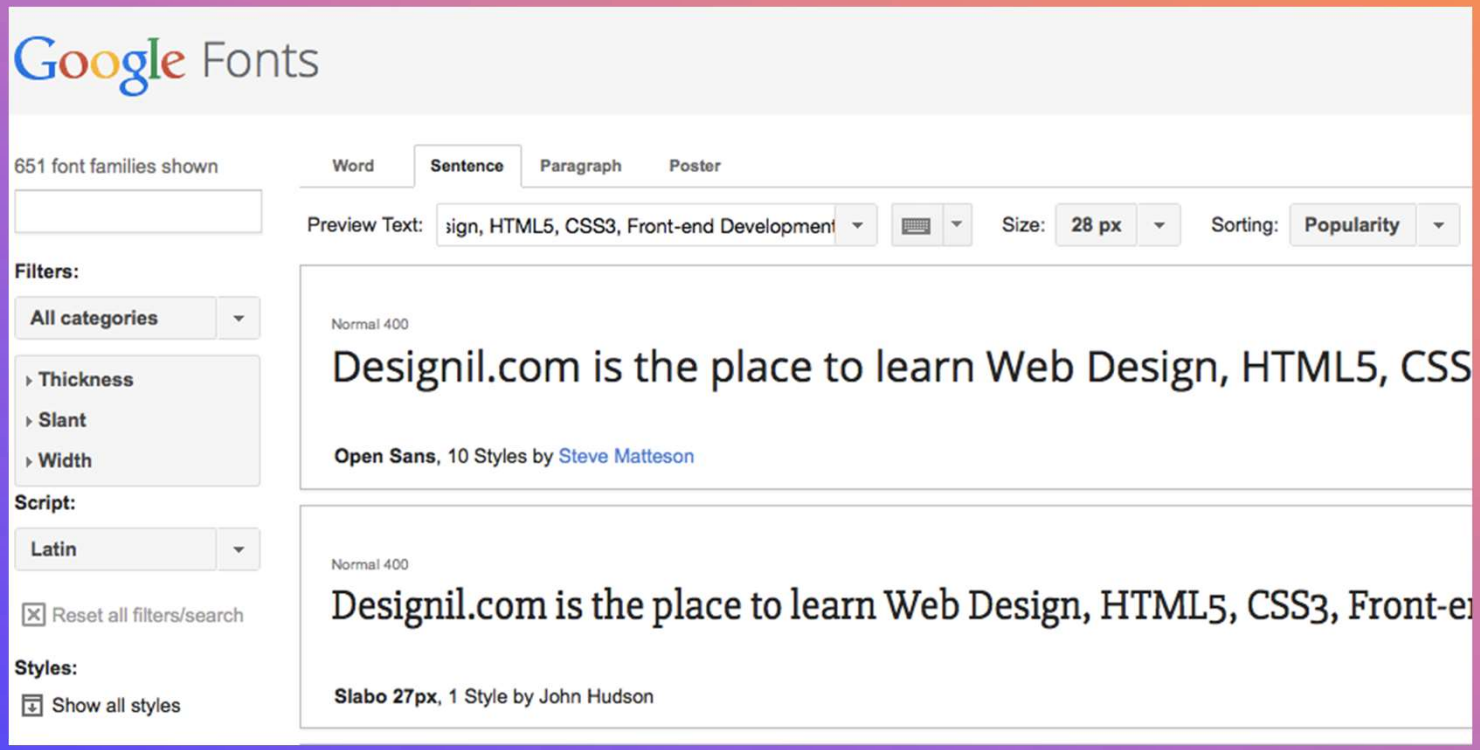
เด่นขึ้น

ตัวหนังสือจุดนั้นมองเห็นได้ชัดเจนกว่าจุดอื่น ไม่ว่าจะเป็นการทำตัวหนังสือใหญ่, ฟ้อนต์หนา, ตัวพิมพ์ใหญ่ ฯลฯ

ส่วนการทำให้ “เด่นน้อยลง” เป็นการทำให้ตัวหนังสือจุดนั้นมองเห็นได้ยากกว่าจุดสำคัญอื่น ไม่ว่าจะเป็นการทำให้ตัวหนังสือเล็ก, ฟ้อนต์บาง, ลด Contrast ฯลฯ



เลือกใช้ฟ้อนต์ให้เหมาะสม

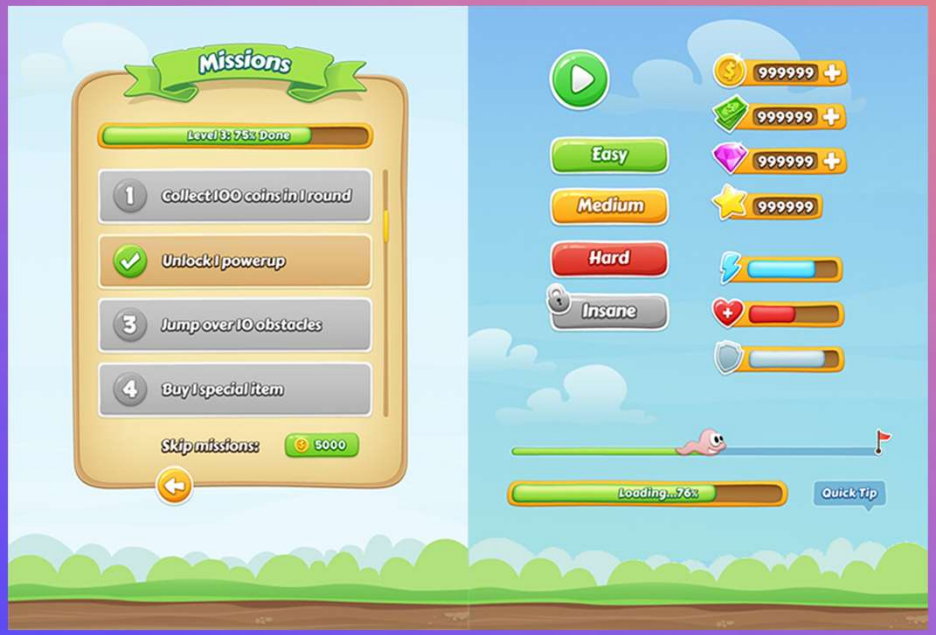



move
54
1,123,456


score
12550
200
50
pause
chat
200
300
50
50
70
30
70
10
400

ขโมยอย่างศิลปิน

ดีไซน์เนอร์ที่เก่ง ย่อมต้องผ่านการเป็น ดีไซน์ที่ไม่เก่ง มาก่อน และการลองทำดีไซน์ตามคนอื่นเพื่อศึกษาก็เป็นวิธีที่ดีที่สุดวิธีหนึ่ง (แต่ไม่ใช่ว่าลอกงานเค้าแล้วไปบอกว่าคิดเอง)



ถายทอด จะเปนเรองของการทบลอกตก
ลงมาจากฟ้าและต้องลงล็อกที่เหมาะสม
(tertis)หรือ การเดินทางในดินแดนแปลก
ประหลาด (Machinarium)

ไม่ใช่ทุกส่วนของเกมที่จะเป็นส่วนหนึ่ง
ของการเล่าเรื่อง ตัวอย่างเช่น game
menu และ HUD (Head up display)
เพราะว่าตัวละครในเกมไม่ต้องสนใจ
องค์ประกอบพวกนี้ แต่ก็ไม่ได้หมายความ
ว่ามันไม่ได้มีส่วนช่วยในการเล่าเรื่องนะ
อย่างพวกเกมอนาคตโดยทั่วไปจะมี ส่วน
ของGUI ที่เป็นแนวอนาคตปรากฏอยู่

ออกแบบด้วย การทำให้ผู้เล่นจม
อยู่กับเกม

และความสมจริงอาจจะไม่ใช่อะไร
ที่คุณต้องการ

และมันอาจจะไปขัดกับการเล่า
เรื่องได้

ถึงแม้จะถูกวิพากย์วิจารณ์ว่ามันเป็น interface ที่ยุ่งยาก แต่จำไว้ว่า World of Warcraft เป็นเกมซับซ้อนที่ให้มีการเล่นที่หลากหลาย ความซับซ้อนของ interface เป็น ผลมาจากความซับซ้อนของเกม

เกี่ยวกับส่วนประกอบในเกม แม้ว่าข้อมูลนั้นจะไม่ได้เป็นส่วนหนึ่งของการเล่าเรื่อง

ตัวอย่างที่ดีก็อย่างเช่น ไอคอนที่ปรากฏอยู่เหนือหัวของตัวละครในThe Sims

บนจออย่าง กระจกแตก หรือเลือดที่มันกระเด็นมาโดน

เอฟเฟคที่โต้ตอบกับ fourth wall เป็นตัวอย่างที่พบบ่อย

ที่สุด

ส่วนประกอบนี้หวังที่จะให้ผู้เล่นรู้สึก

สมจริงโดยทำให้เหมือนว่าเกมมีการ

โต้ตอบกับผู้เล่นโดยตรง ยกตัวอย่างเช่น

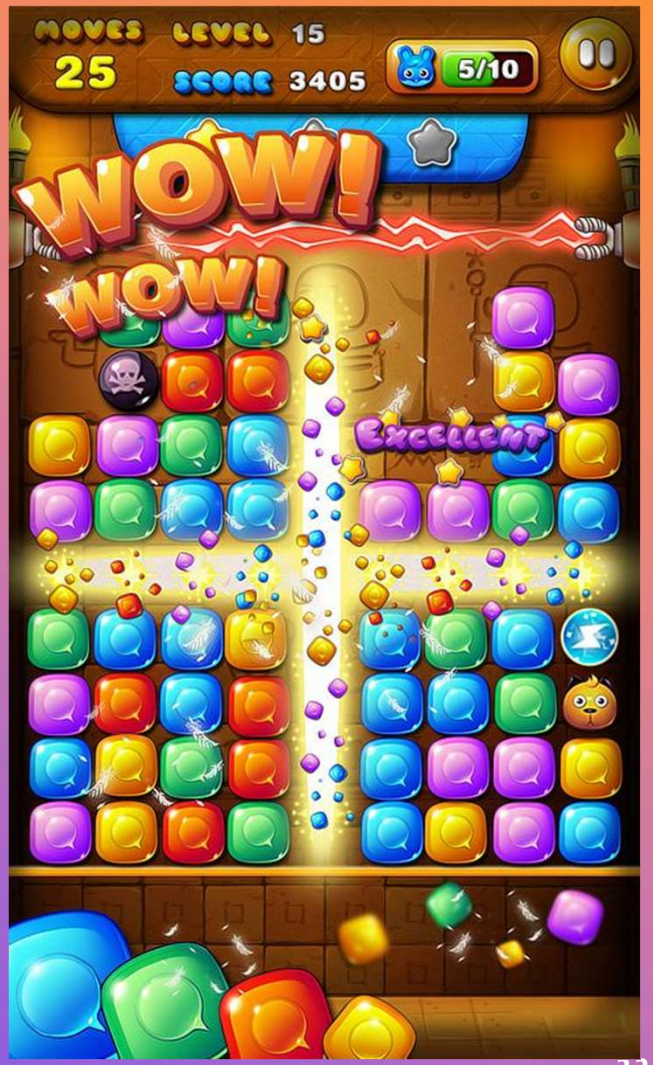
เลือดที่กระเด็นมาติดหน้าจอใน

Killzone2จำไว้ว่าส่วนประกอบ interface

นี้ยังมีผลต่อการลดขอบเขตการมองเห็น

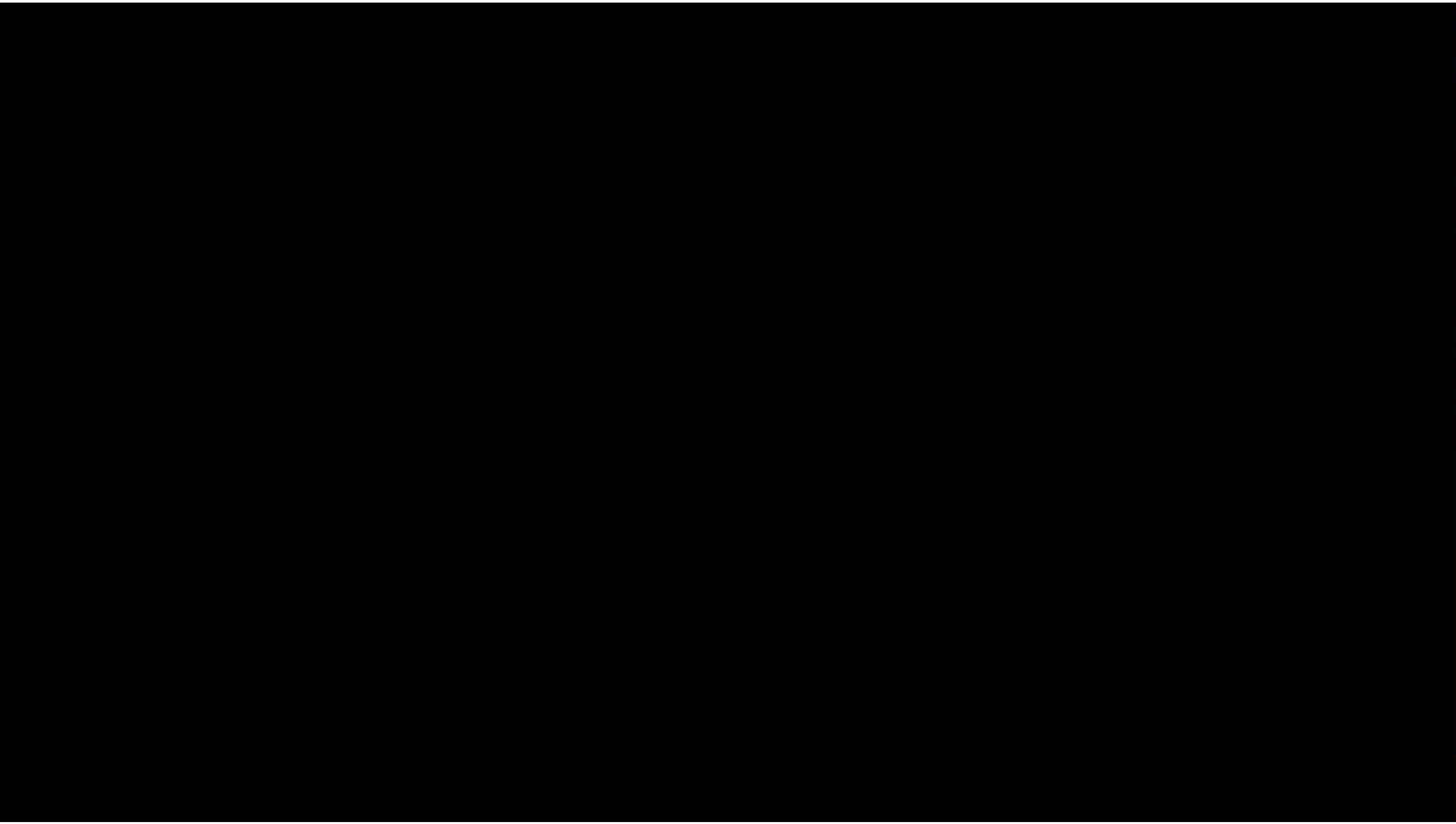
ด้วย


AIRCRAFT LANDED: 00019
00054


MOVES
25
LEVEL 15
SCORE 3405
5/10
WOW!
WOW!
EXCELLENT

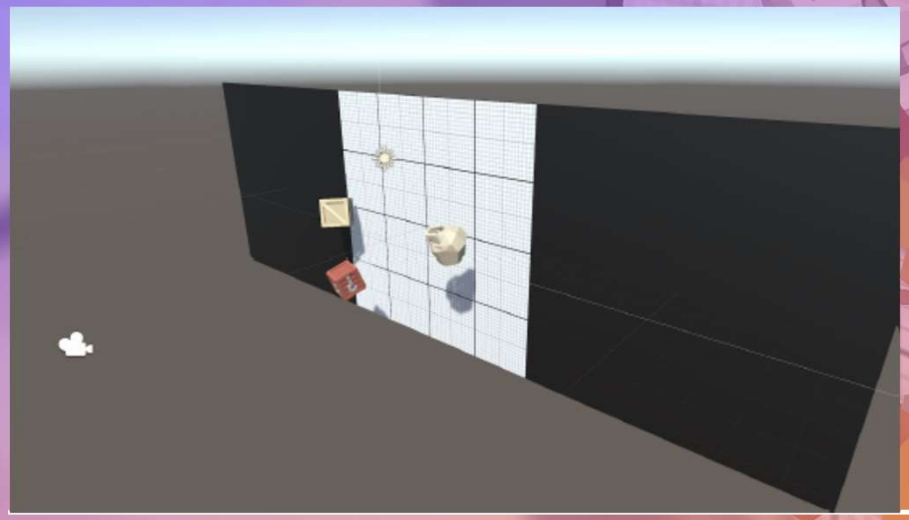
UNIT 5 – USER INTERFACE

(Quick Click Prototype)



Unit 4 – Gameplay Mechanics

- Quick Click Prototype
  - Clicky Mouse
  - Keeping Score
  - Game Over
  - What's the Difficulty?



Clicky Mouse – คลิ๊กๆๆ เอาเมาส์มาคลิ๊ก

Clicky Mouse

- Step 1 : Create project and switch to 2D view
- Step 2 : Create good and bad targets
- Step 3 : Toss objects randomly in the air
- Step 4 : Replace messy code with new methods
- Step 5 : Create object list in Game Manager
- Step 6 : Create a coroutine to spawn objects
- Step 7 : Destroy target with click and sensor

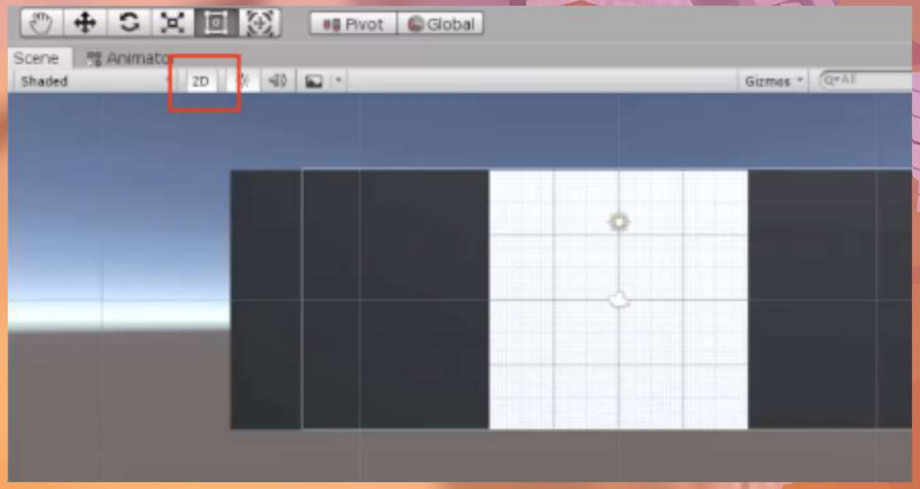
view

One last time… we need to create a new project and download the starter files to get things up and running.

1. Open Unity Hub and create an empty “Prototype 5” project in your course directory on the correct Unity version.
2. Click to download the Prototype 5 Starter Files, extract the compressed folder, and then import the .unitypackage into your project.
3. Open the Prototype 5 scene, then delete the sample scene without saving
4. Click on the 2D icon in Scene view to put Scene view in 2D
5. (optional) Change the texture and color of the background and the color of the borders

New Concept : 2D View

Demo : Notice in 2D view: You can’t rotate around objects or move them in the Z direction



Clicky Mouse – Step 2 : Create good and bad targets

The first thing we need in our game are three good objects to collect, and one bad object to avoid. It'll be up to you to decide what's good and what's bad.

1. From the Library, drag 3 "good" objects and 1 "bad" object into the Scene, rename them "Good 1", "Good 2", "Good 3", and "Bad 1"
2. Add Rigid Body and Box Collider components, then make sure that Colliders surround objects properly
3. Create a new Scripts folder, a new "Target.cs" script inside it, attach it to the Target objects
4. Drag all 4 targets into the Prefabs folder to create "original prefabs", then delete them from the scene

Tip : The bigger the collider boxes, the easier it will be to hit them

Tip : Try selecting multiple objects and applying scripts/components - very handy

Clicky Mouse – Step 3 : Toss objects randomly in the air

Now that we have 4 target prefabs with the same script, we need to toss them into the air with a random force, torque, and position.

1. In Target.cs, declare a new private Rigidbody targetRb; and initialize it in Start()
2. In Start(), add an upward force multiplied by a randomized speed
3. Add a torque with randomized xyz values
4. Set the position with a randomized X value

New Function : AddTorque

Tip : Test with different values by dragging them in during runtime

Don't worry : We're going to fix all these hard-coded values next

```
private Rigidbody targetRb;

void Start() {
  targetRb = GetComponent<Rigidbody>();
  targetRb.AddForce(Vector3.up * Random.Range(12, 16), ForceMode.Impulse);
  targetRb.AddTorque(Random.Range(-10, 10), Random.Range(-10, 10),
  Random.Range(-10, 10), ForceMode.Impulse);
  transform.position = new Vector3(Random.Range(-4, 4), -6); }
```

methods

Instead of leaving the random force, torque, and position making our Start() function messy and unreadable, we're going to store each of them in brand new clearly named custom methods.

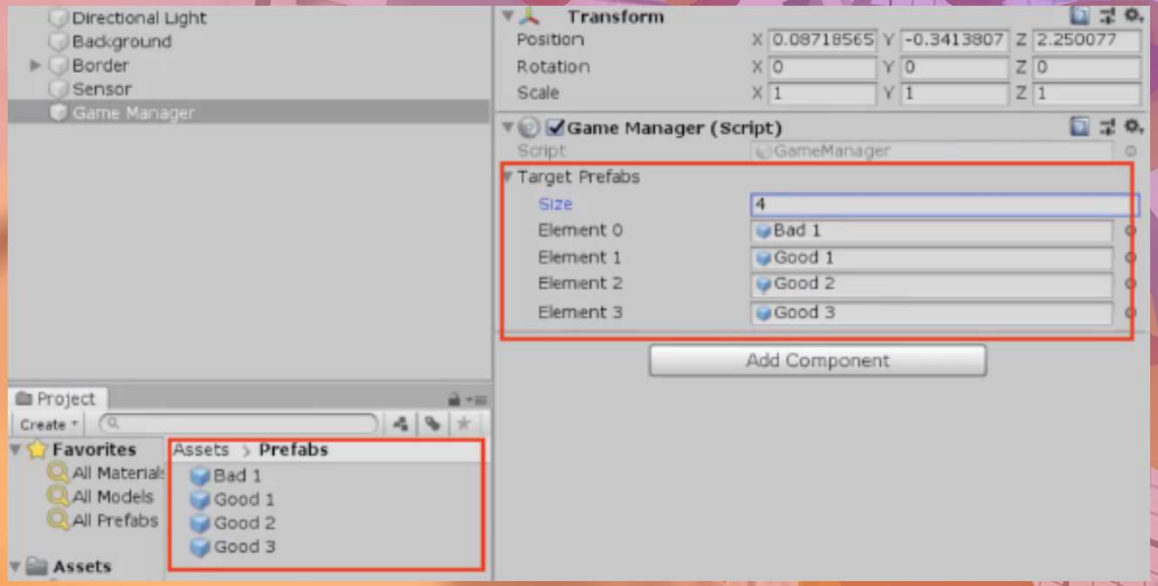
1. Declare and initialize new private float variables for minSpeed, maxSpeed, maxTorque, xRange, and ySpawnPos;
2. Create a new function for Vector3 RandomForce() and call it in Start()
3. Create a new function for float RandomTorque() and call it in Start()
4. Create a new function for RandomSpawnPos(), have it return a new Vector3 and call it in Start()

```
private float minSpeed = 12;
private float maxSpeed = 16;
private float maxTorque = 10;
private float xRange = 4;
private float ySpawnPos = -6;

void Start() {
  ...
  targetRb.AddForce(... RandomForce(), ForceMode.Impulse);
  targetRb.AddTorque(... RandomTorque(), RandomTorque(), RandomTorque(),
    ForceMode.Impulse);
  transform.position = ... RandomSpawnPos();
}

Vector3 RandomForce() {
  return Vector3.up * Random.Range(minSpeed, maxSpeed);
}
float RandomTorque() {
  return Random.Range(-maxTorque, maxTorque);
}
Vector3 RandomSpawnPos() {
  return new Vector3(Random.Range(-xRange, xRange), ySpawnPos);
}
```

Manager

The next thing we should do is create a list for these objects to spawn from. Instead of making a Spawn Manager for these spawn functions, we're going to make a Game Manager that will also control game states later on.

1. Create a new "Game Manager" Empty object, attach a new GameManager.cs script, then open it
2. Declare a new public List<GameObject> targets;, then in the Game Manager inspector, change the list Size to 4 and assign your prefabs



New Concept : Lists

New Concept : Game Manager

Demo : Feel free to reference old code: We used an array instead of a list to spawn the animals in Unit 2

objects

Now that we have a list of object prefabs, we should instantiate them in the game using coroutines and a new type of loop.

1. Declare and initialize a new private float spawnRate Variable
2. Create a new IEnumerator SpawnTarget () method
3. Inside the new method, while(true), wait 1 second, generate a random index, and spawn a random target
4. In Start(), use the StartCoroutine method to begin spawning objects

```
private float spawnRate = 1.0f;

void Start() {  StartCoroutine(SpawnTarget());  }

IEnumerator SpawnTarget() {
  while (true) {
    yield return new WaitForSeconds(spawnRate);
    int index = Random.Range(0, targets.Count);
    Instantiate(targets[index]); } }
```

Tip : Feel free to reference old code:
we used coroutines for the powerup cooldown in Unit 4
Tip : Arrays return an integer with .Length, while Lists return an integer with .Count
New Concept : While Loops

sensor

Now that our targets are spawning and getting tossed into the air, we need a way for the player to destroy them with a click. We also need to destroy any targets that fall below the screen.

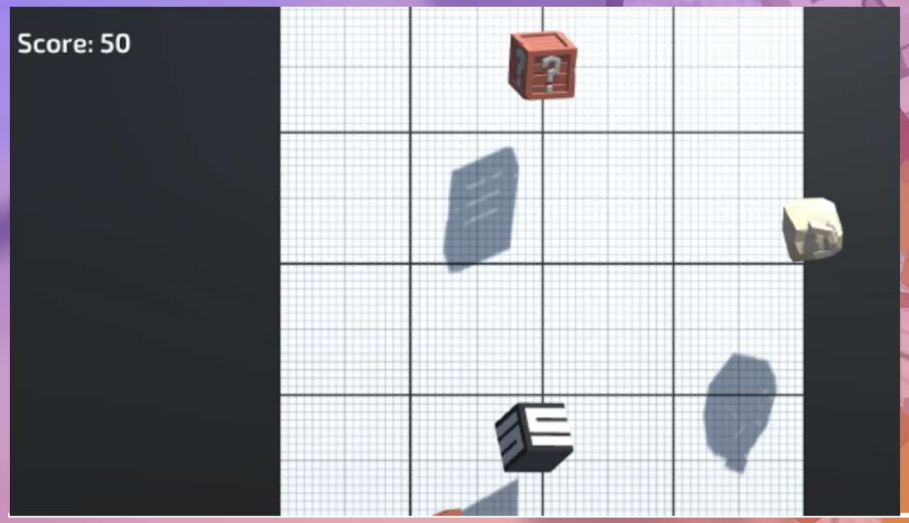
1. In Target.cs, add a new method for private void OnMouseDown() { }, and inside that method, destroy the gameObject
2. Add a new method for private void OnTriggerEnter(Collider other) and inside that function, destroy the gameObject

```
private void OnMouseDown() {
  Destroy(gameObject); }

private void OnTriggerEnter(Collider other) {
  Destroy(gameObject); }
```

New Function : OnMouseDown

Tip : There is also OnMouseUp, and OnMouseEnter, but Down is definitely the one we want

Tip : You could use Update and check if target y position is lower than a certain value, but a sensor is better because it doesn't run all the time



Keeping Score – เก็บคะแนนเอาไว้ในความทรงจำ

Keeping Score

- Step 1 : Add Score text position it on screen
- Step 2 : Edit the Score Text's properties
- Step 3 : Initialize score text and variable
- Step 4 : Create a new UpdateScore method
- Step 5 : Add score when targets are destroyed
- Step 6 : Assign a point value to each target
- Step 7 : Add a Particle explosion

2D view

In order to display the score on-screen, we need to add our very first UI element.

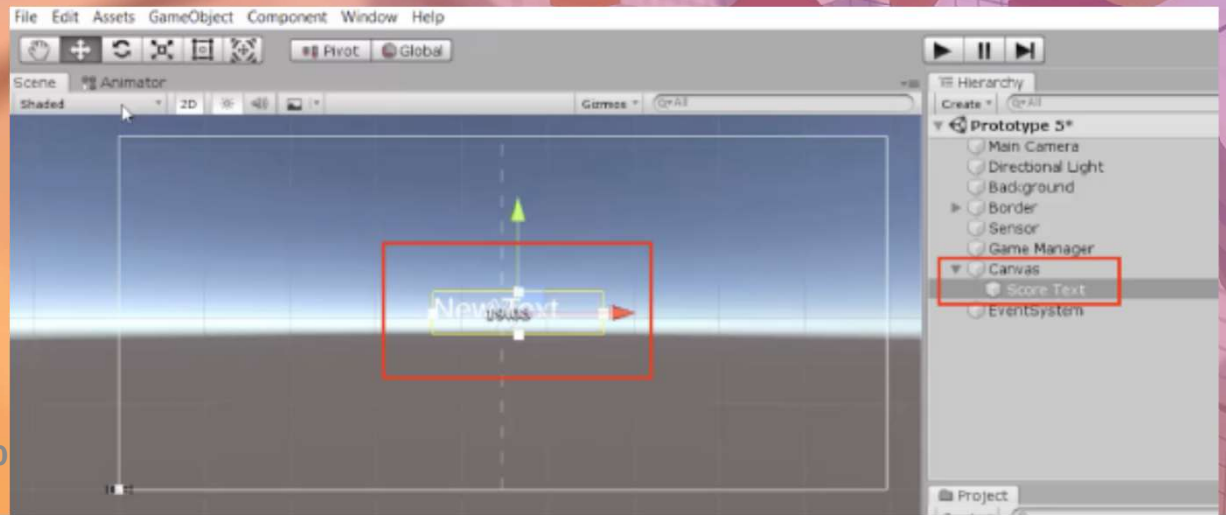
1. In the Hierarchy, Create > UI > TextMeshPro text, then if prompted click the button to Import TMP Essentials
2. Rename the new object “Score Text”, then zoom out to see the canvas in Scene view
3. Change the Anchor Point so that it is anchored from the top-left corner
4. In the inspector, change its Pos X and Pos Y so that it is in the top-left corner

New Concept : Text Mesh Pro / TMPro

New Concept : Canvas

New Concept : Anchor Points

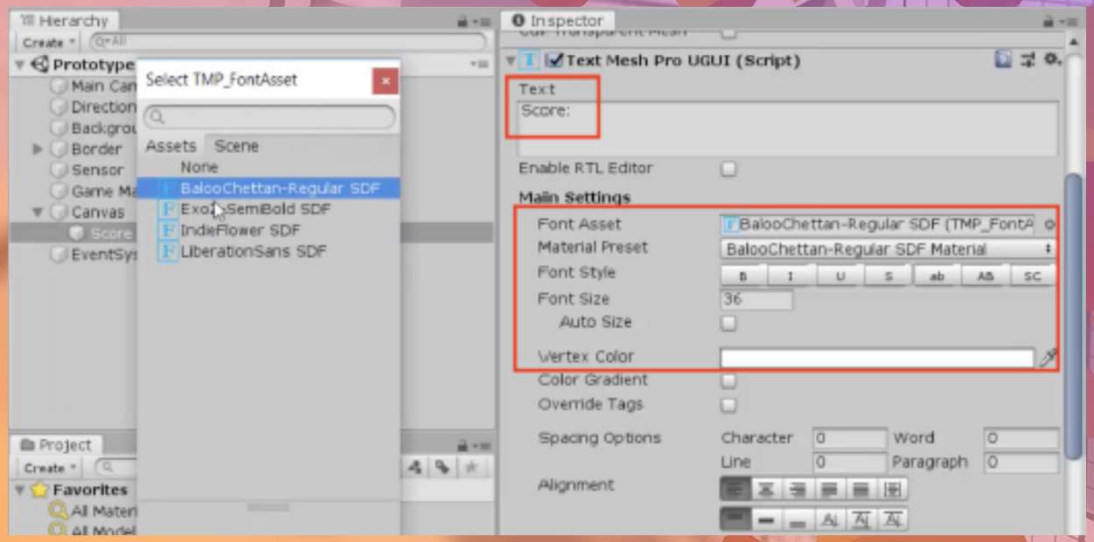
Tip : Look at how it displays in scene vs game view. It may be hard to see white text depending on the background



properties

Now that the basic text is in the scene and positioned properly, we should edit its properties so that it looks nice and has the correct text.

1. Change its text to “Score:”
2. Choose a Font Asset, Style, Size, and Vertex color to look good with your background



variable

We have a great place to display score in the UI, but nothing is displaying there! We need the UI to display a score variable, so the player can keep track of their points.

1. Change its text to "Score:"
2. Declare a new public TextMeshProUGUI scoreText, then assign that variable in the inspector
3. Create a new private int score variable and initialize it in Start() as score = 0;
4. Also in Start(), set scoreText.text = "Score: " + score;

New Concept : Importing Libraries

```
using TMPro;

private int score;
public TextMeshProUGUI scoreText;

void Start() {
  StartCoroutine(SpawnTarget());
  score = 0;
  scoreText.text = "Score: " + score; }
```

Keeping Score - Step 1: Create a new UpdateScore method

The score text displays the score variable perfectly, but it never gets updated. We need to write a new function that racks up points to display in the UI.

1. Create a new private void UpdateScore method that requires one int scoreToAdd parameter
2. Cut and paste scoreText.text = "Score: " + score; into the new method, then call UpdateScore(0) in Start()
3. In UpdateScore(), increment the score by adding score += scoreToAdd;
4. Call UpdateScore(5) in the spawnTarget() function

New Concept : Custom functions requiring parameters

Don't worry : It doesn't make sense to add to score when spawned, this is just temporary

```
void Start() {
  ...scoreText.text = "Score: " + score;
  UpdateScore(0); }

IEnumerator SpawnTarget() {
  while (true) {  ... UpdateScore(5);  }

private void UpdateScore(int scoreToAdd) {
  score += scoreToAdd;
  scoreText.text = "Score: " + score; }
```

Keeping Score – Step 3: Add score when targets are destroyed

Now that we have a method to update the score, we should call it in the target script whenever a target is destroyed.

1. In GameManager.cs, make the UpdateScore method public
2. In Target.cs, create a reference to private GameManager gameManager;
3. Initialize GameManager in Start() using the Find() method
4. When a target is destroyed, call UpdateScore(5);, then delete the method call from SpawnTarget()

Tip : Feel free to reference old code: We used script communication in Unit 3 to stop the game on GameOver

Warning : If you try to call UpdateScore while it's private, it won't work

GameManager.cs

```
IEnumerator SpawnTarget() {
  while (true) {  ... UpdateScore(5);  }

private public void UpdateScore(int scoreToAdd) { ... }
```

Target.cs

```
private GameManager gameManager;

void Start() {
  ... gameManager = GameObject.Find("Game Manager")
        .GetComponent<GameManager>();}

private void OnMouseDown() {
  ... gameManager.UpdateScore(5); }
```

Keeping Score – Step 5: Assign a point value to each target

The score gets updated when targets are clicked, but we want to give each of the targets a different value. The good objects should vary in point value, and the bad object should subtract points.

1. In Target.cs, create a new public int pointValue variable
2. In each of the Target prefab's inspectors, set the Point Value to whatever they're worth, including the bad target's negative value
3. Add the new variable to UpdateScore(pointValue);

```
public int pointValue;

private void OnMouseDown() {
  Destroy(gameObject);
  gameManager.UpdateScore(5 pointValue); }
```

Tip : Here's the beauty of variables at work. Each target $ can have their own unique pointValue!

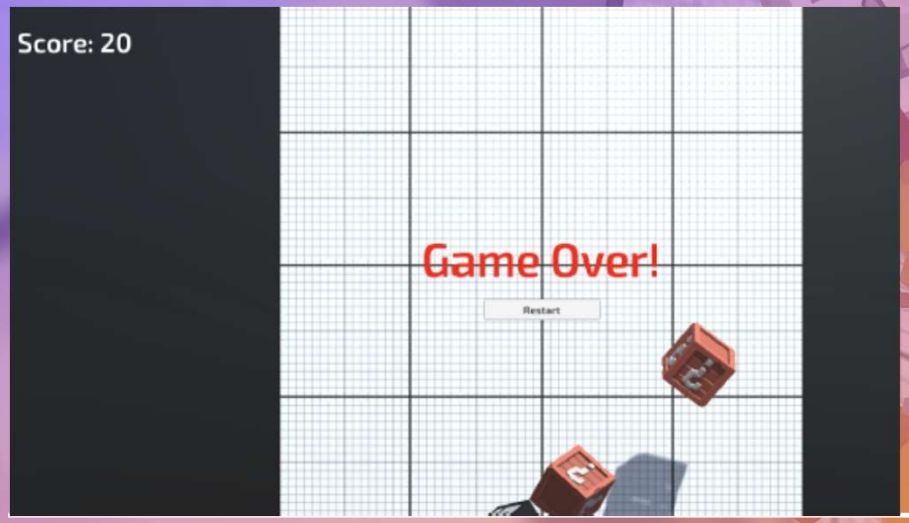
Keeping Score – Step 7 : Add a Particle explosion

The score is totally functional, but clicking targets is sort of… unsatisfying. To spice things up, let's add some explosive particles whenever a target gets clicked!

1. In Target.cs, add a new public ParticleSystem explosionParticle variable
2. For each of your target prefabs, assign a particle prefab from Course Library > Particles to the Explosion Particle variable
3. In the OnMouseDown() function, instantiate a new explosion prefab

```
public ParticleSystem explosionParticle;

private void OnMouseDown() {
  Destroy(gameObject);
  Instantiate(explosionParticle, transform.position,
  explosionParticle.transform.rotation);
  gameManager.UpdateScore(pointValue); }
```

ติดตามตอนต่อไป



Game Over – เจ้าน่ะ...ได้แพ้แล้วววว