การเขียนโปรแกรมคอมพิวเตอร์ขั้นสูงเพื่อควบคุมอุปกรณ์

Advance Computer Programming

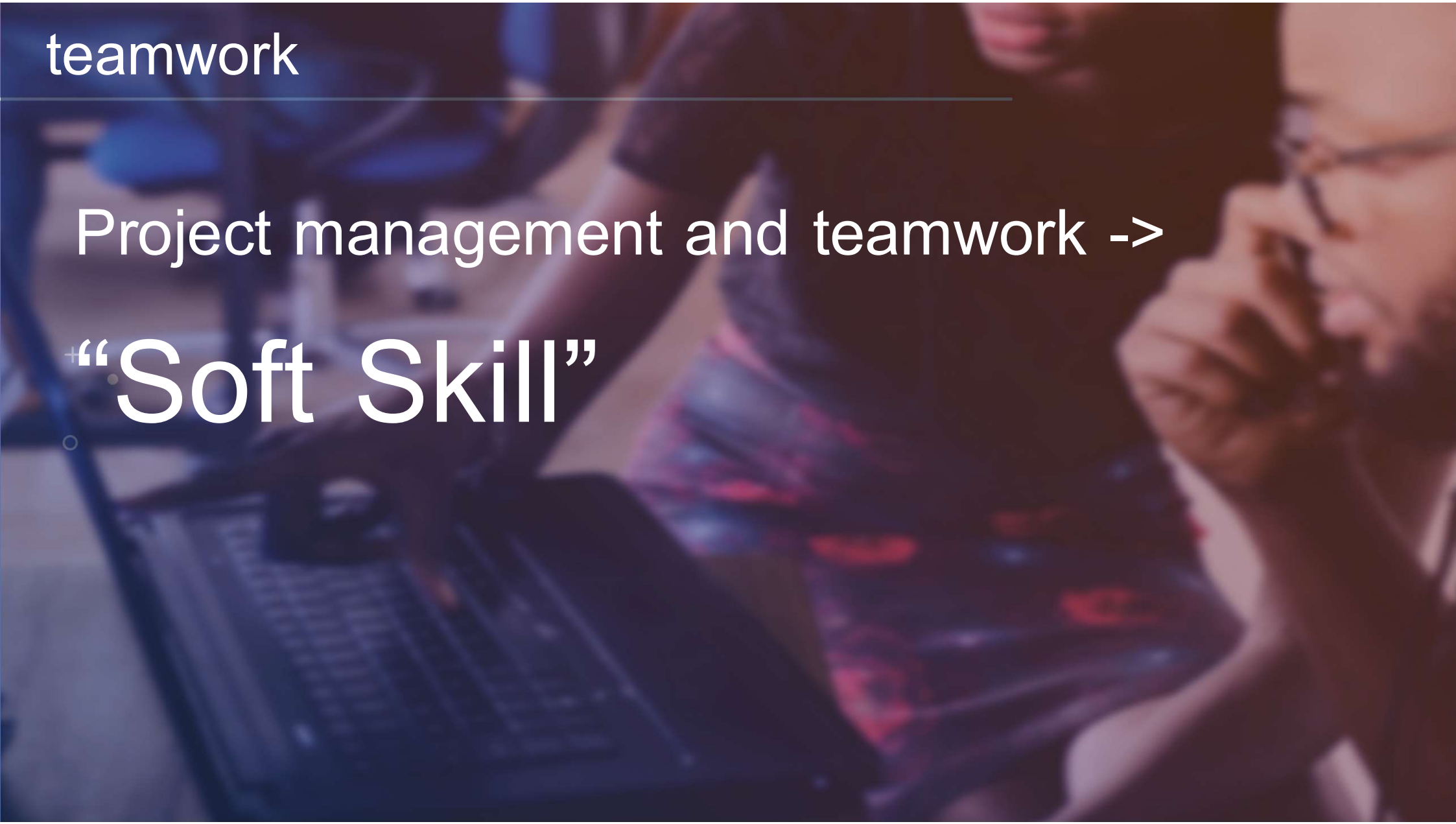
[สัปดาห์ที่ 4]

สอนโดย พงศธร เกียรติเจริญพร (มิว)

Introduction to project management and teamwork

teamwork

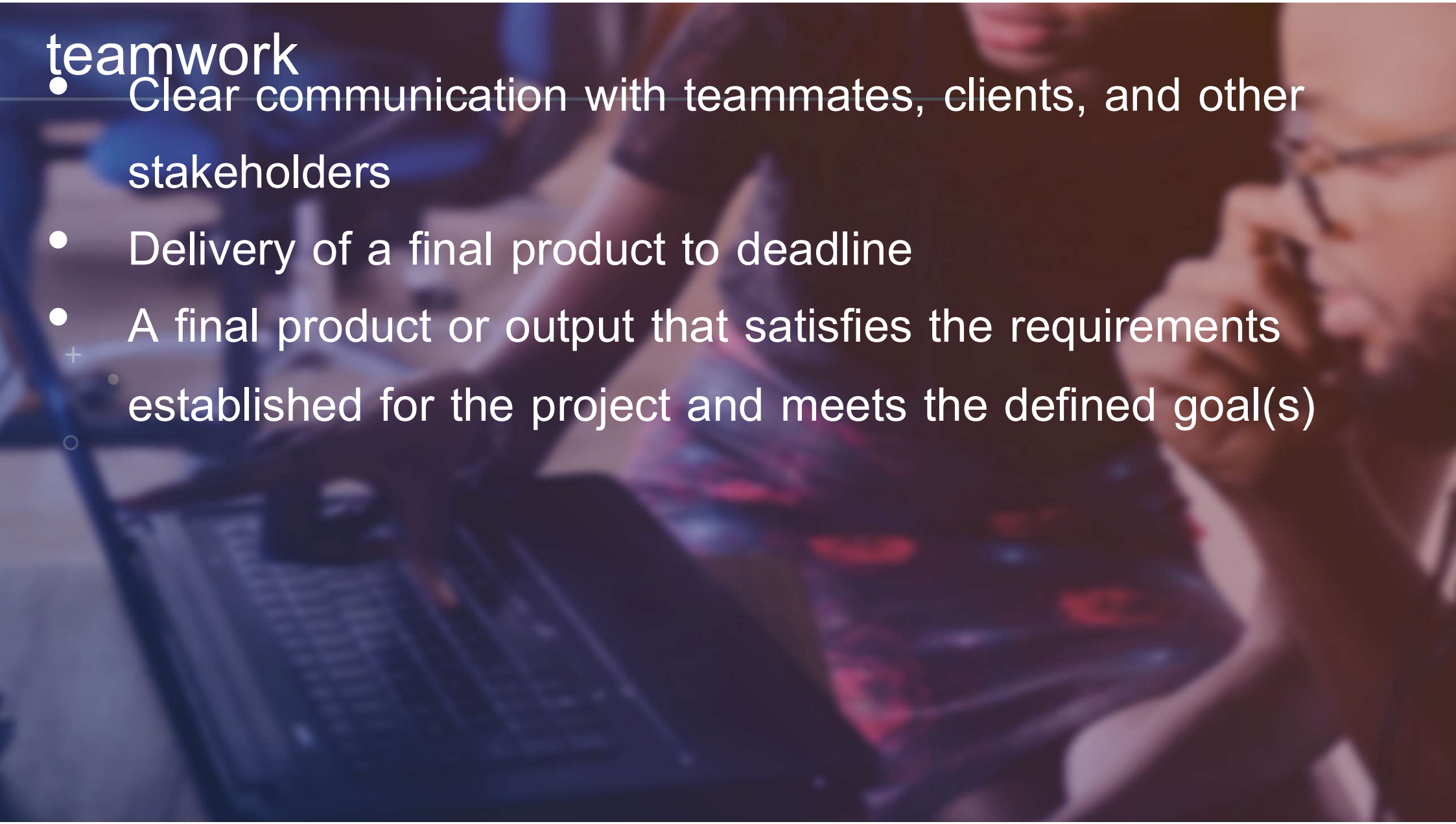
- Overview
- Phases of production
  - Overview
  - The pre-production phase
  - The production phase
  - The post-production phase
  - The operations phase
  - Unity throughout production
- Overview of project planning
- Design documents and project plans
- Managing projects and tracking progress



teamwork

Project management and teamwork ->

“Soft Skill”



teamwork

- Clear communication with teammates, clients, and other stakeholders
- Delivery of a final product to deadline
- A final product or output that satisfies the requirements established for the project and meets the defined goal(s)

Phases of production

Pre-production → Production → Post-production → Operations

- **Preproduction:** k do -sc production begins, such as planning, prototyping, pipeline setup, and initial designs..

- **Production:** Creation of the product and assets within it, including the creation of final 2D images and 3D models, audio, lighting, and user experience.

Phases of production

Pre-production → Production → Post-production → Operations

- **Post-production** one pro to be complete, including quality assurance (QA), editing, testing, bug fixing, and final polishing.

- **Operations** : Ongoing work after a product has been released to keep it running, such as sales, monetization, updates, and continued maintenance.

Phases of production – Video Overview

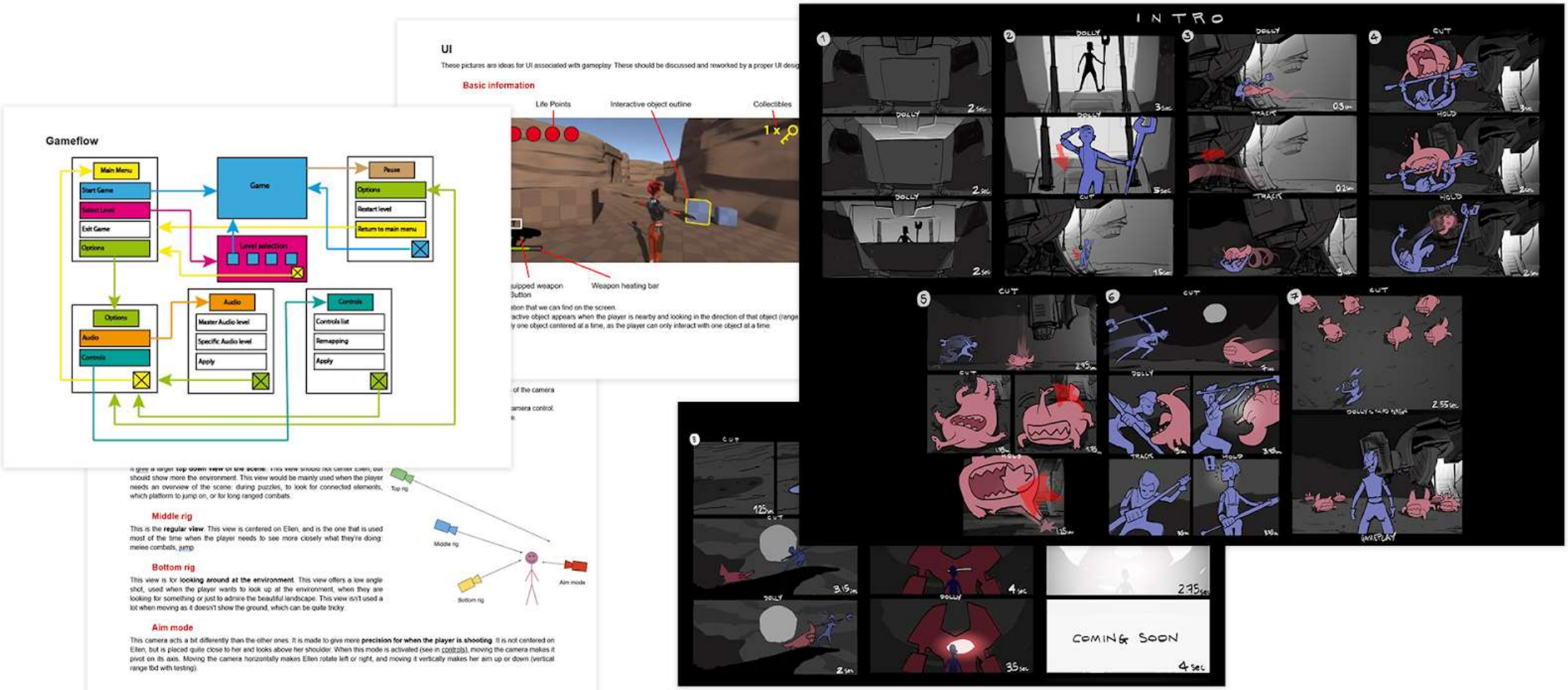
Pre-production → Production → Post-production → Operations

Pre-production Phase

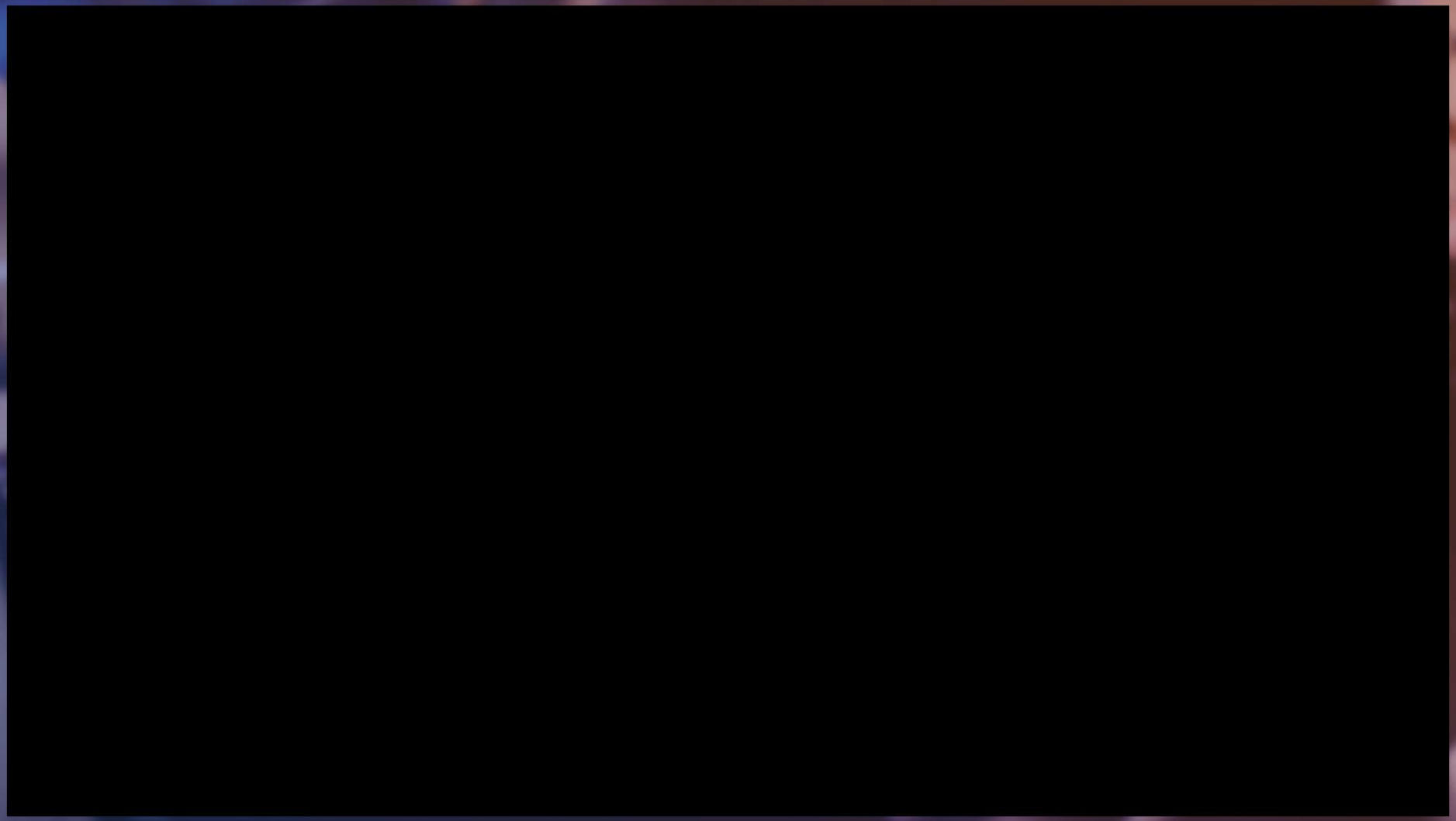
ในช่วงของ Pre-Pr

studios produce a design document for their product: a single source of truth for its creative direction. For a film or animation, this may take the form of a script and storyboards, depicting the contents, look, and feel of each scene. For a game, a game design document includes information about the story, gameplay, art direction, intended target audience and accessibility.

Pre-production Phase



Pre-production Phase - Video



Pre-production ช่วงถามตอบ

Imagine your studio is making a game, an animation, or a simulation using a real-time editor like Unity — but you decide to skip pre-production entirely and jump right into production.

Imagine a disaster scenario that could have been avoided if you had taken necessary time in pre-production.

How could pre-production have solved these problems?

Production Phase

Pre-production → **Production** → Post-production → Operations

Production is typically the longest and most expensive phase of product development, when all artists, developers, managers, and directors come together to create the actual product. With all the big decisions already made during pre-production, content developers begin churning out assets and content to bring the ideas from the design document to life on screen.

production?

Pre-production → **Production** → Post-production → Operations

Programming	Art	System Design	Audio	Management
Software developers	3D artists	Game Designers	Audio designer	Producers
Mobile developers	2D artists	Level Designers	Sound engineer	Product owners
Virtual / Augmented Reality developers	Lighting artists	Systems architect	Composer	Creative directors
Artificial intelligence engineers	Character artists	Script writer		QA managers
Graphics engineers	Technical artists	Narrative designer		
Network engineers	Visual Effects artists	User Experience (UX) Designer		
Gameplay programmers	Animators			
Quality assurance technicians	Cinematographers			
	User Interface (UI) Designer			

Production Phase

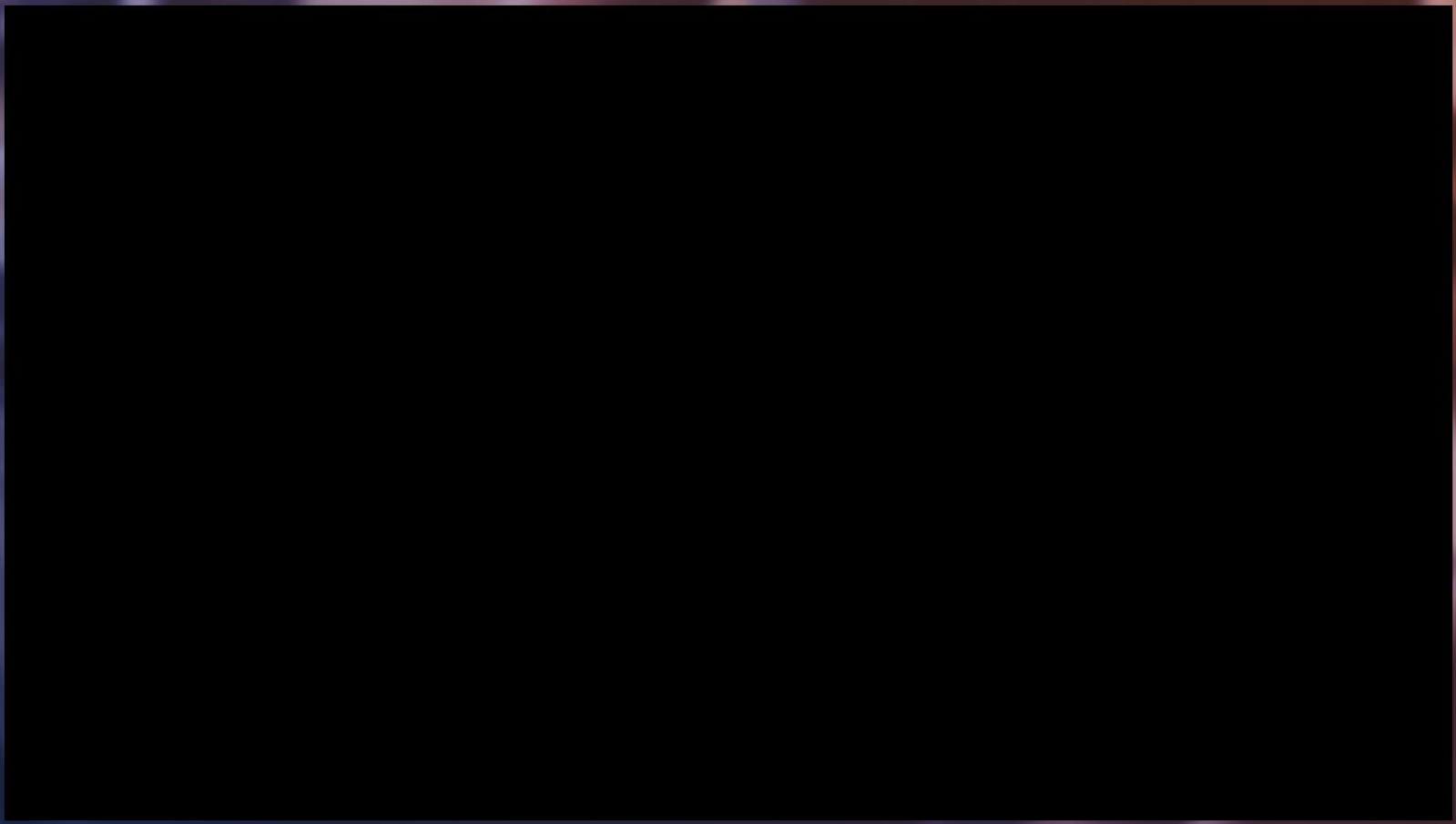
Pre-production → **Production** → Post-production → Operations

Given all of the moving parts and people during production, it is critical to have a **producer** overseeing the development of the entire project and making sure the team is hitting critical deadlines. The producer also has the important job of preventing **feature creep**, the alluring but risky tendency to add shiny new features to the product rather than stick to the ones agreed to during pre-production.

The production phase ends on a deadline, which should have been defined during pre-production.

Production Phase – Video 1/2

Pre-production → **Production** → Post-production → Operations



Production Phase - Version control

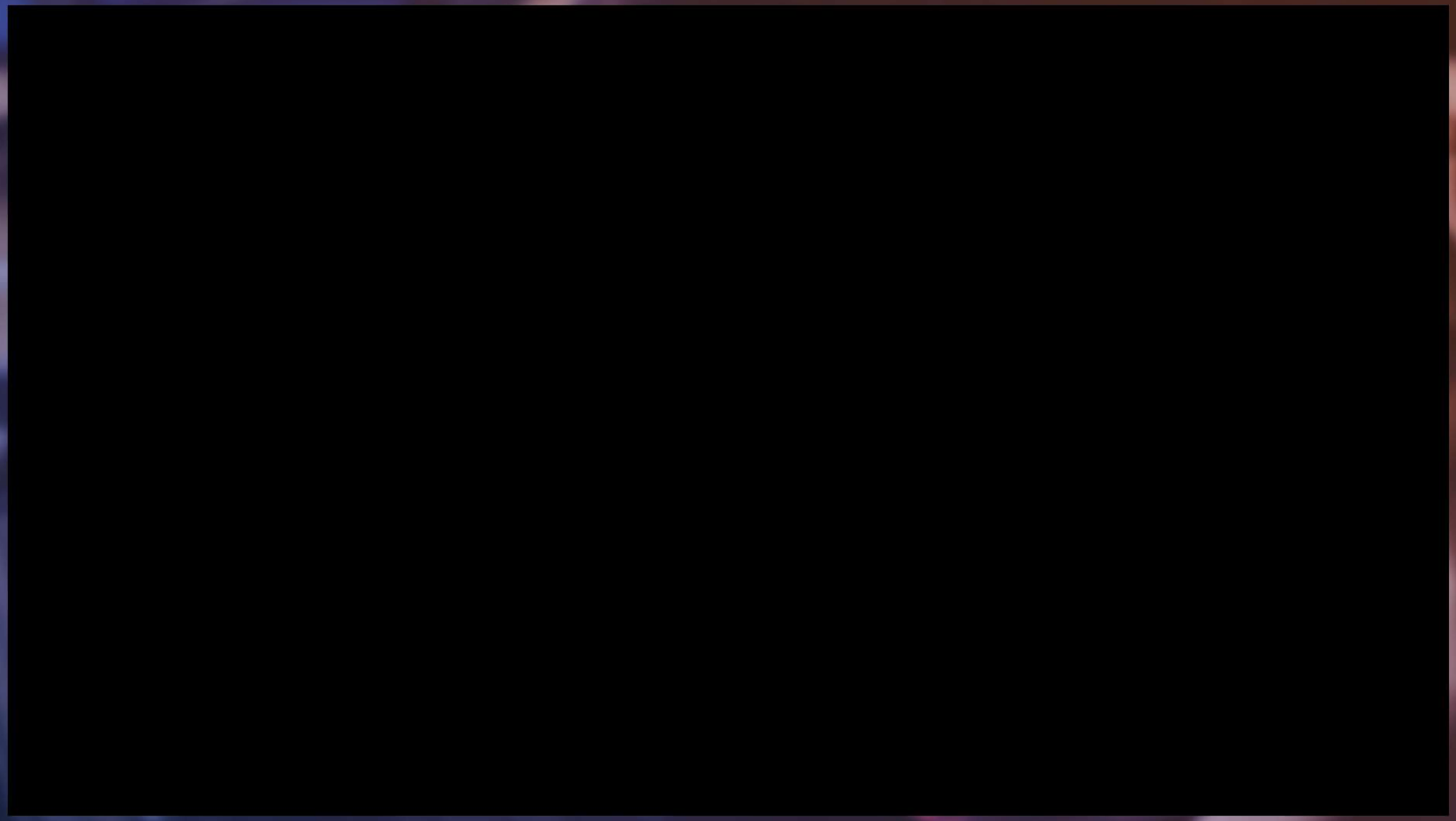
One tricky aspect of production is version control, the complex process of managing and integrating new content and code from team members across departments. When you are working on a team — and even if you're working solo — it is critical to keep track of every modification to the many files in a project so that you can track your progress and go back to earlier versions as needed. Popular version control software includes Github, Unity's Collaborate, or PlasticSCM.

Production Phase - Version control

Teams use these version control systems to:

- Track where changes to the project have been made over time.
- Revert back to a previous version of the project, if necessary.
- Manage contributions from all team members.

Production Phase – Video 2/2



Post-Production Phase

Pre-production → Production → **Post-production** → Operations

Post-production begins when the project is technically complete, but not yet ready for release

Post-Production Phase

What happens during post-production?

Pre-production → Production → Post-production → Operations

In post-production, the project is evaluated, edited, polished, and fixed. This usually includes **alpha testing** and **beta testing**. Alpha testing is performed in-house to identify issues and areas for improvement, whereas beta testing is carried out by potential end-users in the expected environment where the product will be used. Both processes typically produce long lists of bugs and feature requests that are prioritized and addressed in the post-production phase.

Post-Production Phase

The Operations Phase

What is operations?

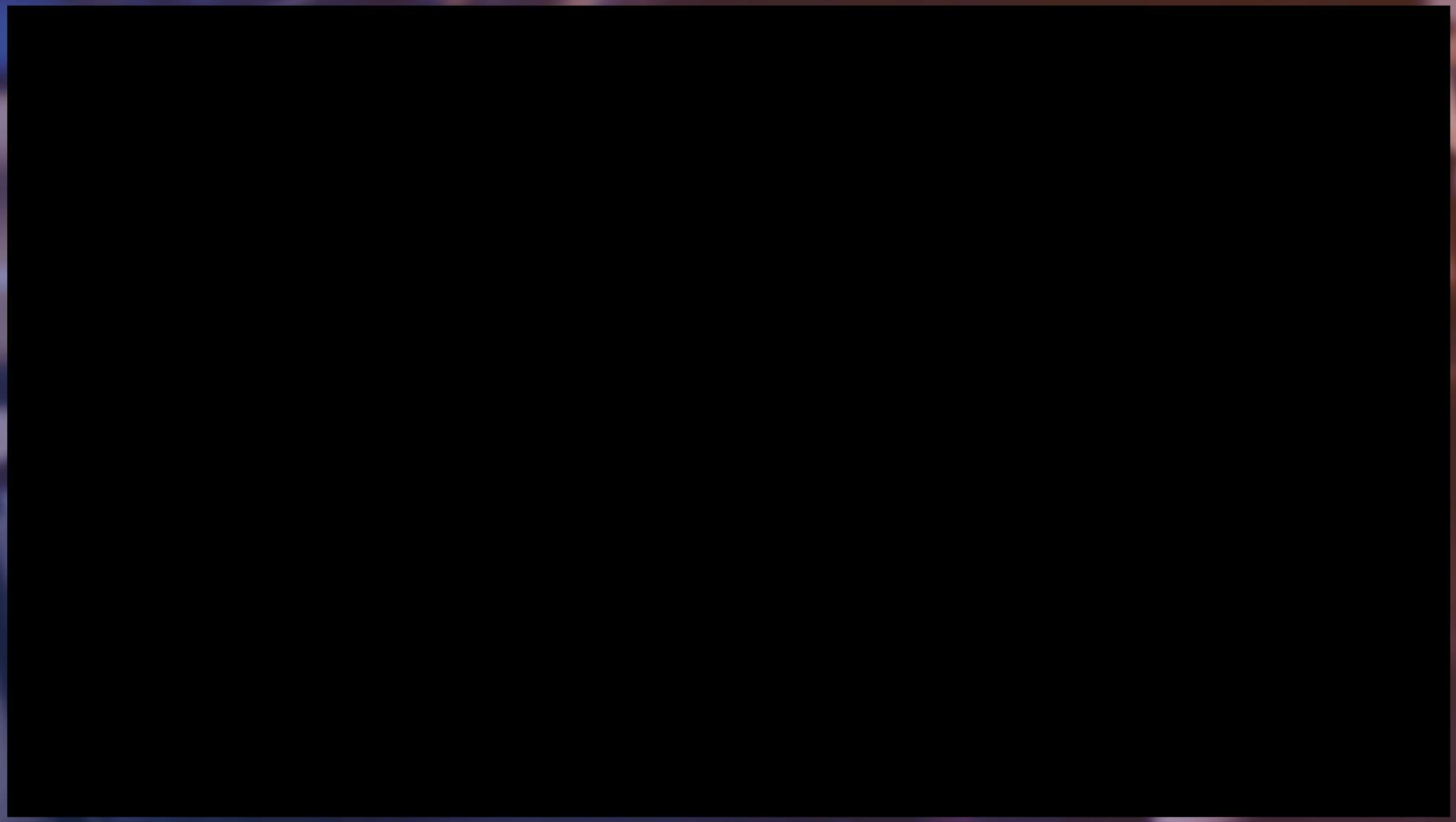
After a product has been released it enters the operations phase, which includes all the work required to keep it running such as sales, analytics, monetization, updates, and continued maintenance. The operations phase looks completely different in each industry.

operations?

- Support: dealing with incoming questions, requests, and problems from

bring in revenue
- Analytics: tracking and analyzing user data to inform any needed changes to the game's functionality, marketing, or business strategy
- Server maintenance: ensuring that any servers supporting the game are operating properly
- Website maintenance: managing and updating the website that promotes or sells the game
- Business and sales: continued marketing, public relations, partnerships, and other strategies to promote sales of the game

The Operations Phase - Video



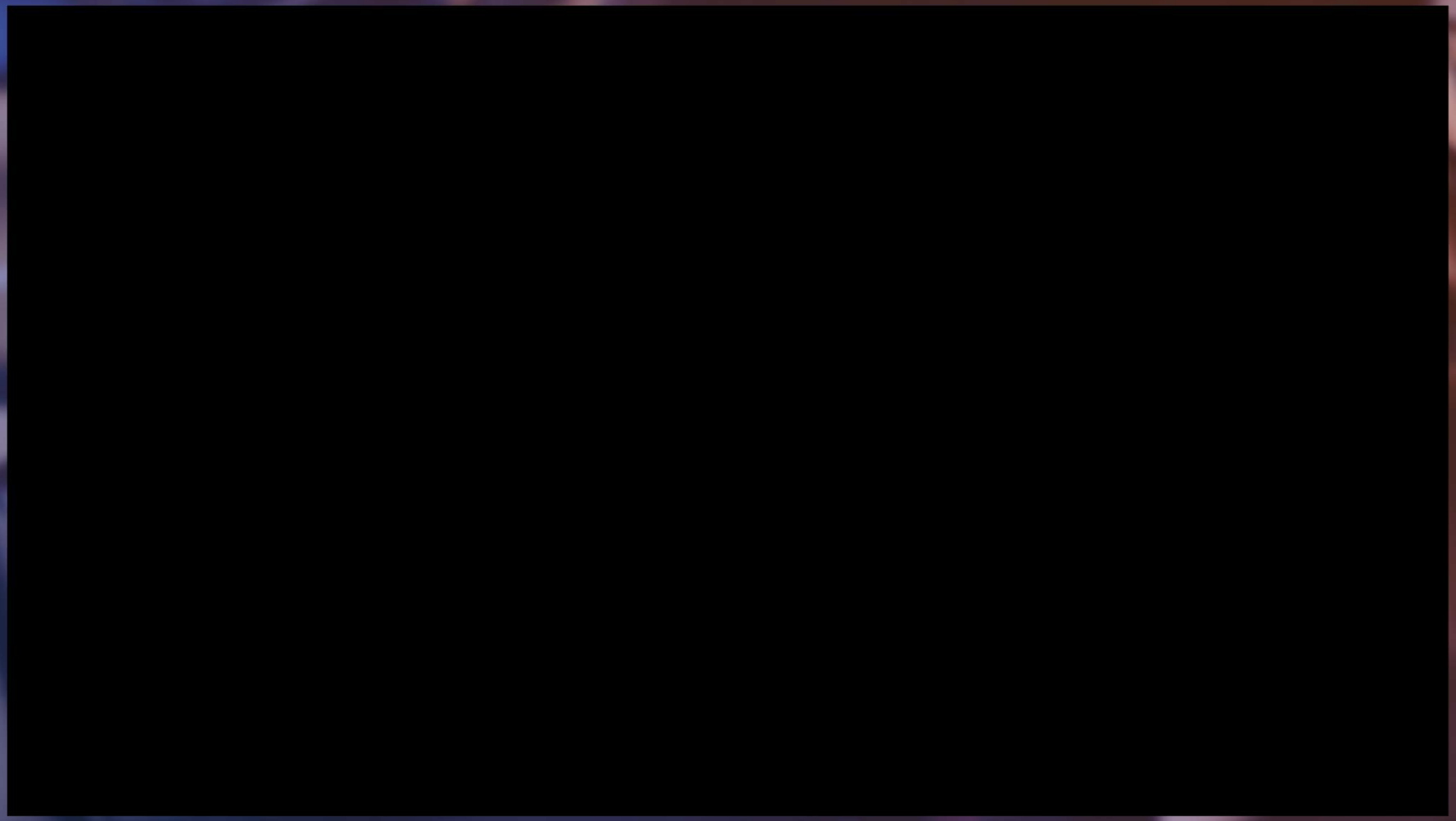
Unity throughout production

- Pre-production, to create concepts, prototypes, or visualize camera angles.

How is Unity used during the production cycle?

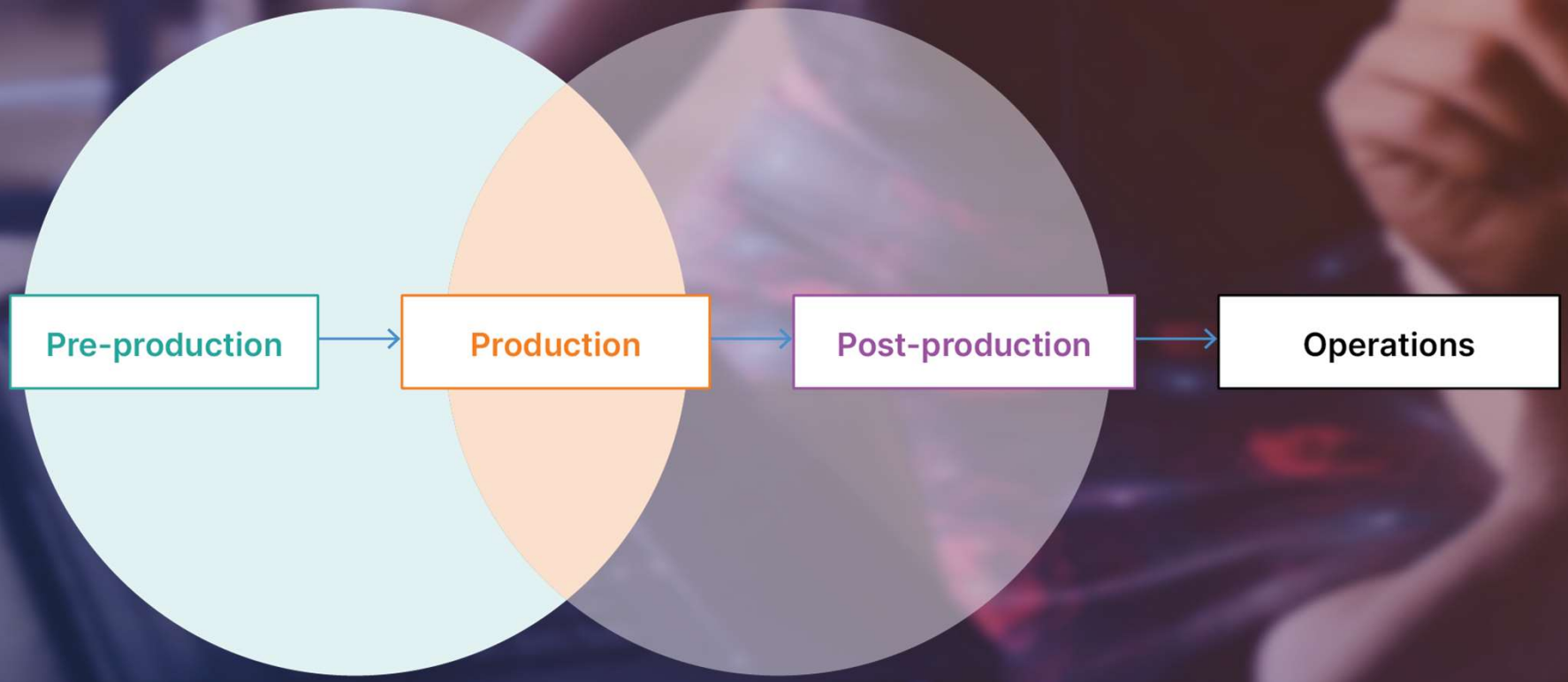
- Production, to create the project itself in a real-time environment, integrating art, audio, and code.
- Post-production, to test the overall functionality, analyze and optimize project performance, and export for various mediums and platforms.
- Operations, to monetize the product through advertising and in-app purchases, address ongoing bug fixes, and push updates to improve or expand upon the product.

Unity throughout production - Video



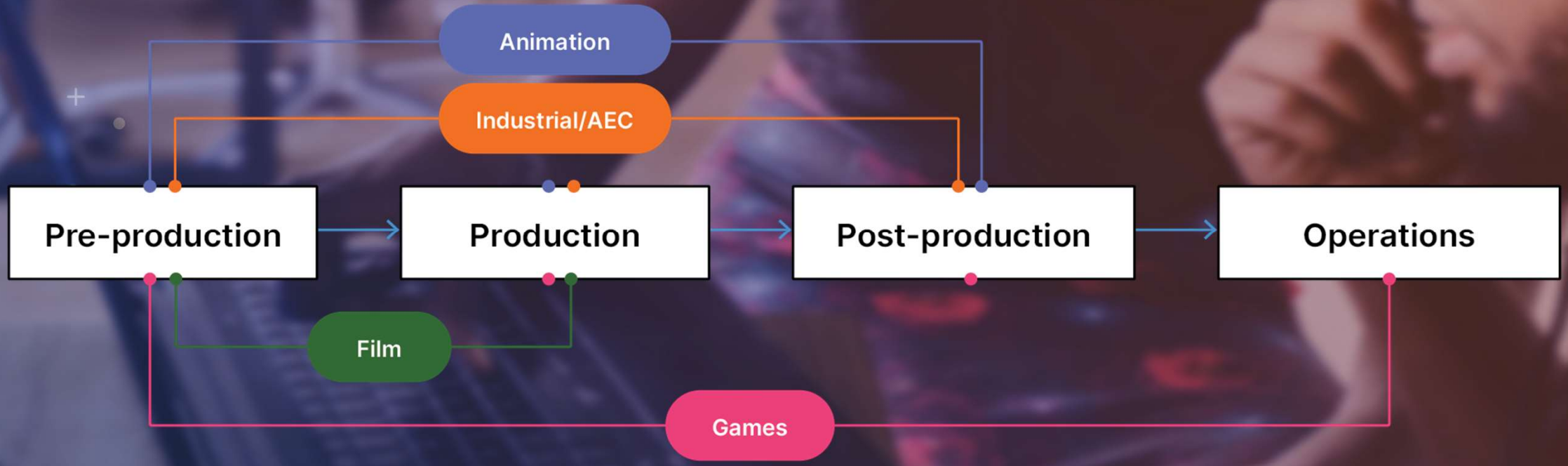
# of production

The advantage of using a real-time engine like Unity is that it blurs the lines between the traditionally linear phases of production by providing immediate feedback on functionality, lighting, textures, and camera angles.

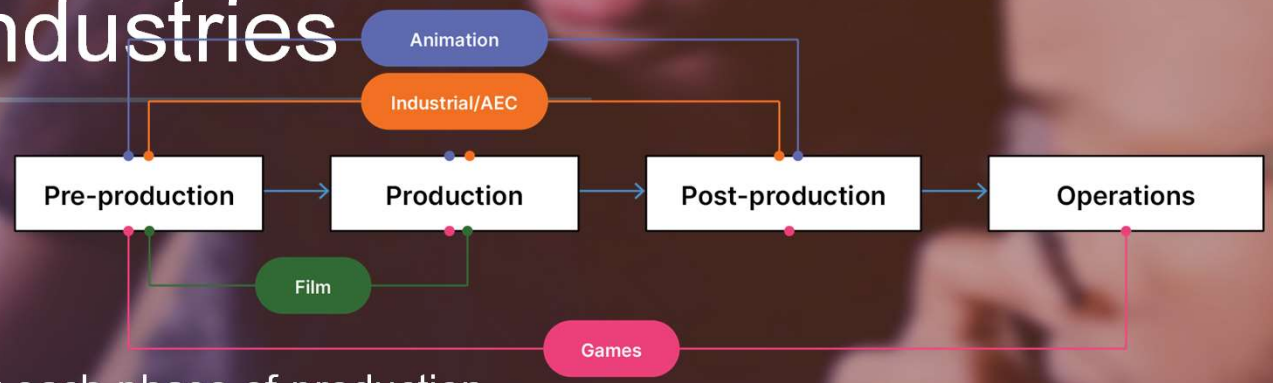


Unity’s role between industries

Unity is used across industries such as games, media and entertainment, AEC (architecture, engineering, and construction), and ATM (automotive, transportation, and manufacturing), but the precise way Unity fits into the production cycle in any of those industries varies.



Unity’s role between industries



- In gaming, Unity is heavily present throughout each phase of production, from pre-production right through to operations.
- In animation, Unity can be used from pre-production concept work right through production of the animation itself and the finishing touches during post-production.
- In film, Unity may be most helpful in pre-production and in production, where it allows studios to much more quickly prototype and visualize ideas in real time.
- In AEC and ATM, Unity is used heavily during pre-production for visualizations and planning. It is also used throughout the full production cycle to make VR and AR applications that help stakeholders visualize how a physical product would appear in the real world.

guide the design and development process:

tracking

- Identify the purpose, audience, and goals of your project.
- Identify the necessary project steps by creating a project plan.
- Within the project plan, create a timeline with specific deliverables and due dates.
- Consistently track your milestones in order to produce deliverables and meet deadlines.
- Assign roles when working in teams, and define and prioritize tasks for you and all teammates.
- Make sure you and all teammates are following up and following through on roles and responsibilities.

Overview of project planning - **Communication**

When communicating with others on your project:

- Be clear about your progress and any issues that impact the work of others, whether they are members of your team or supporting external collaborators
- Be respectful of others' time — and of your own.
- When critiquing work, remain constructive and sensitive to the feelings of others. Focus on making your feedback helpful, specific, and respectful.
- Be open to feedback yourself by actively listening and engaging with the person delivering the feedback. Reflect honestly on how their feedback can be addressed.

- Be punctual. (ตรงต่อเวลา)

professionalism

- Promptly reply to collaborators, peers, and clients. (ตอบกลับทันท่วงที)
- Listen to others' opinions and contributions. (รับฟังความคิดเห็น)
- Actively engage in collaborative work. (มีส่วนร่วมอย่างกระตือรือร้น)

Design documents and project plans

In the pre-production phase at the start of the project, design documents are created to help define and scope what you are going to create. These documents include:

- Game (or Experience) Design Documents (GDDs)
- Target user personas
- Project charters
- Technical specifications

Design documents

Design documents contain the blueprint for your project. They include:

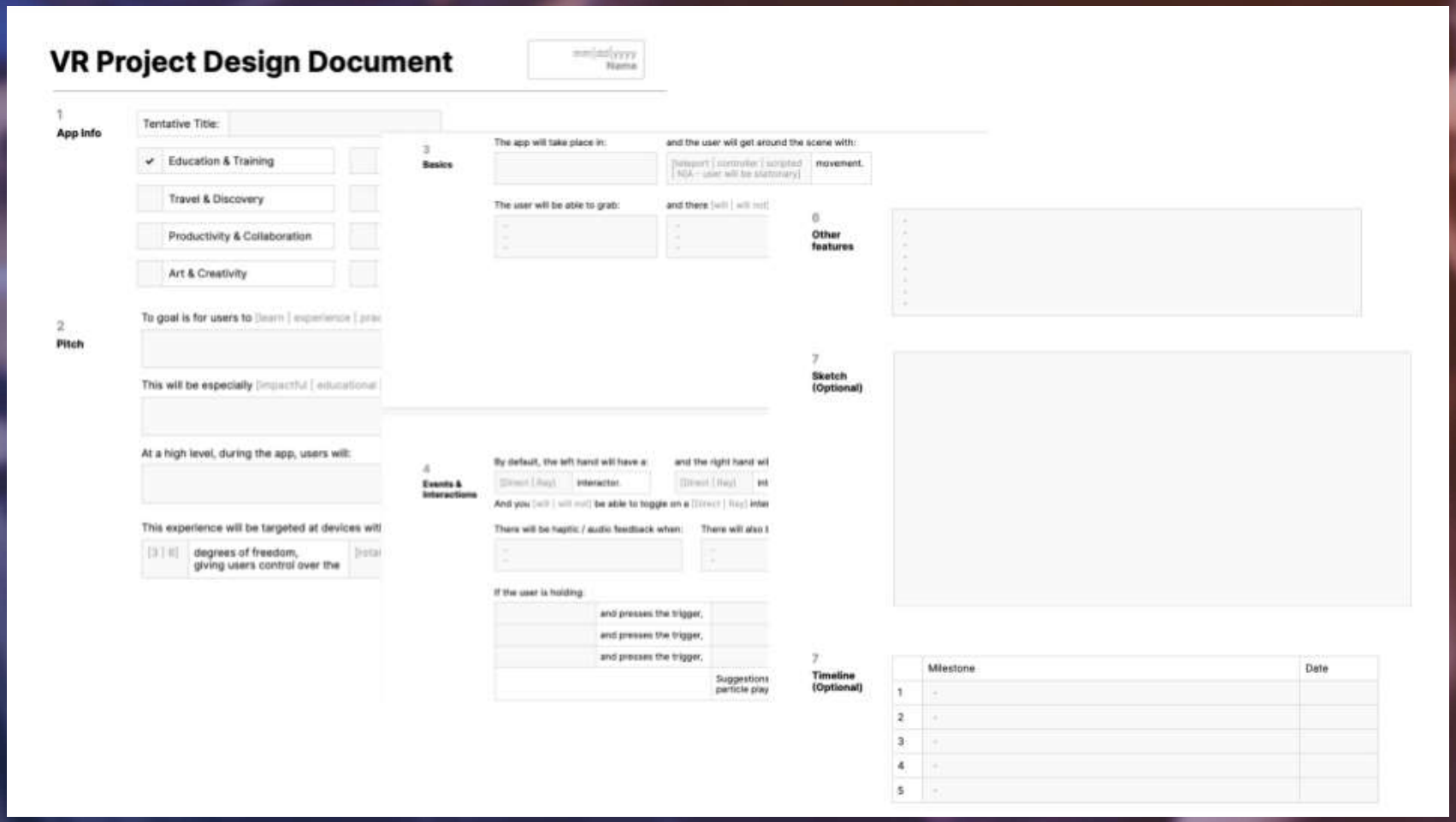
- High level overviews; for example, an overall project vision in the Game (or Experience) Design Document
- Requirements and standards for particular pipelines in the project
- Detailed design specifications for particular features

Design documents

In your high level design document, you should identify:

- The goal and purpose for the project
- The intended users and audience
- Key features of the project
- The final form of delivery

Design documents



VR Project Design Document

1
App Info

Tentative Title:

- ✔ Education & Training
- Travel & Discovery
- Productivity & Collaboration
- Art & Creativity

2
Pitch

To goal is for users to [learn | experience | prac

This will be especially [impactful | educational

At a high level, during the app, users will:

This experience will be targeted at devices wit

[3 | 6] degrees of freedom, giving users control over the [rota

3
Basics

The app will take place in:

and the user will get around the scene with:

[teleport | controller | scripted | N/A - user will be stationary] movement.

The user will be able to grab:

and there [will | will not]

4
Events & Interactions

By default, the left hand will have a:

[Direct | Ray] interactor.

and the right hand wi

[Direct | Ray] int

And you [will | will not] be able to toggle on a [Direct | Ray] inter

There will be haptic / audio feedback when:

There will also b

If the user is holding:

and presses the trigger,

and presses the trigger,

and presses the trigger,

Suggestions particle play

6
Other features

7
Sketch (Optional)

7
Timeline (Optional)

	Milestone	Date
1	-	
2	-	
3	-	
4	-	
5	-	

Data Types กับ Variables

[access modifier] [data type] [variable name] [variable value]

Data Types

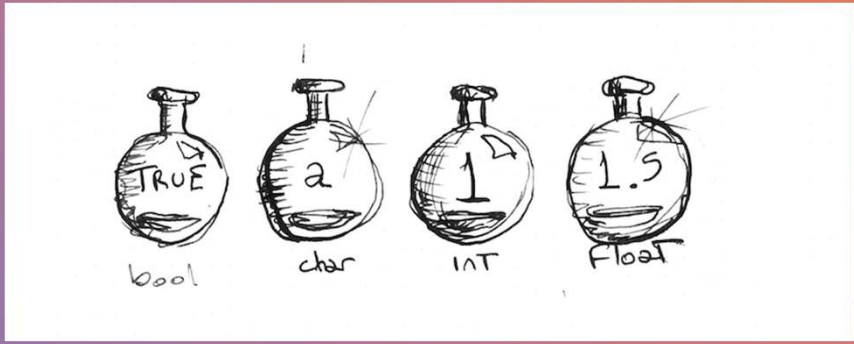
- Integer - จำนวนเต็ม
  -1999 , 0 , -99 , 16 , 300
- Float - จำนวน
  -150.4 , 16.0 , -34.4456 , 9004.55
- String - ตัวอักษร
  "hello" , "Game Over!!" , "400.55" , "yOu TuBE" , "131314"
- Boolean - จริง หรือ เท็จ (ค่าตรรกศาสตร์)
  true , false
- Class , Struct , etc.
  .....

Variable

```
//-----Basic-----
int age = 11;
float height = 134.5f;
string name = "Peter";
bool raining = true;
```

```
//-----Array-----
string[] animals = {"Cat","Dog"};
print(animals[0]);
int[] scores = {7 , 8 , 10};
print(scores[0]+scores[1]+scores[2]);
```

```
//-----Class-----
Monster poring = new Monster();
poring.hp = 50;
poring.exp = 5;
poring.dmg =
Random.Range(5,10);
```



Access modifier

Public vs. Private

```
public float speed;
float turnSpeed = 45.0f;
private float horizontalInput;
private float forwardInput;
```

Choice and Loop

Choice and Loop

Choice (if , if - else)

```
bool raining = true;
if(raining)
{
        print("yes yes yes");
}
else
{
        print("no no no");
}
```

Loop (while loop , for loop)

```
int num = 0;
while(num <= 100)
{
        print(num);
        num = num + 1;
}
```

```
for(int i = 0 ; i <= 100 ; i = i+1)
{
        print(i);
}
```

Condition - เงื่อนไข

Condition

ด้วยเครื่องหมายเปรียบเทียบ

< > <= >= == !=

```
int money = 50000;
bool rich = money > 100000;
if(rich)
{
        print("He's rich!!");
}
else if(money <= 100)
{
        print("He's poor!!");
}
else
{
        print("He's regular person
}
```

1

2

3

Condition (cont.)

____ && ____
____ || ____
!____

- การคำนวณ :
- และ / And / &&
จะเป็นจริงเมื่อทั้ง 2 ฝั่งเป็นจริง
- หรือ / Or / ||
จะเป็นจริงเมื่อฝั่งใดฝั่งหนึ่งเป็นจริง
- ไม่ / Not / !
พลิกความจริง/เท็จ เป็นตรงกันข้าม

ตัวอย่างการใช้งาน

โจทย์

ฉันจะกลับบ้านเมื่อถึงเวลา 1ทุ่ม และ ฝนไม่ตก

- สถานการณ์ไหนถึงจะได้กลับบ้าน
  - 18:50น. ฝนไม่ตก
  - 19:08น. ฝนตก
  - 20:40น. ฝนไม่ตก
  - 17:45น. ฝนตก

```
float time = 18.50f;
bool raining = false;

if(time >= 19.00f && !raining)
{
        print("go home");
}
else
{
        print("stay");
}
```

For loop

For loop

```
for(int i = 0 ; i < 10 ; i = i
+ 1)
{
        print(i);
}
```

```
int i = 0;
while(i < 10)
{
        print(i);
        i = i + 1;
}
```

Scope ของตัวแปร

Scope ของตัวแปร

```
void Start ( )
{
        int a = 10; //ขั้น 1 - local
}
```

```
int a = 10; //ขั้น 2 - global
void Start ( )
{
        print("Start : " + a * 2);
        Hello();
}
void Hello ( )
{
        print("Hello : " + a * 3);
}
```

```
public class MyScript1 : MonoBehaviour
{
        public int a = 10; //ขั้น 3 - public
        void Start ( )
        {
                print(a * 2);
        }
}
//--------อีกไฟล์----------
public class MyScript2 : MonoBehaviour
{
        void Start ( )
        {
                MyScript1 s1 = new MyScript1();
                print(s1.a * 3);
        }
}
```

Function - Input , Output

Function - Input , Output

```
//output    ชื่อฟังก์ชัน        input
void      MyFunction    (          )
{
          //ชุดคำสั่ง
  +       //
     •    //
} °
```

```
int Add (int a , int b)
{
          return a + b;
}

string GetName()
{
          return "Poring";
}
void SayMyName(string n)
{
          print("My name is " + n);
}
```

Session 2 – Basic Gameplay

- Feed the Animals
  - Player Positioning
  - Food Flight
  - Random Animal Stampede
  - Collision Decisions

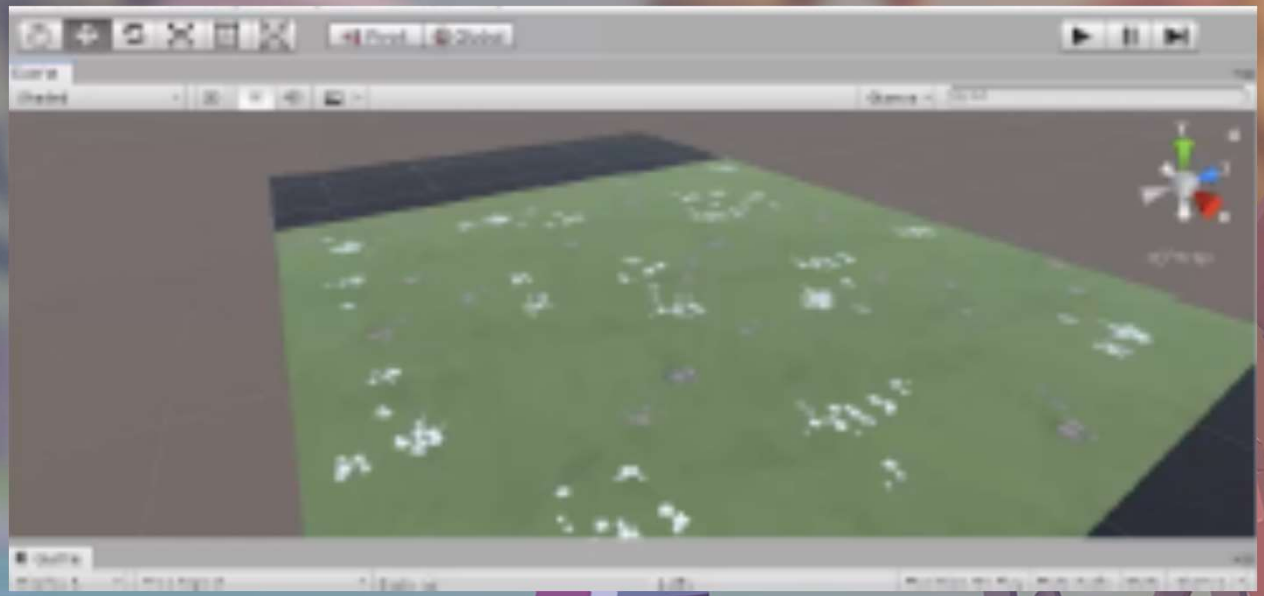
Player Positioning - จัดตำแหน่งตัวละคร

Player Positioning

- Step 1 : Create a new Project for Protype 2
- Step 2 : Add the Player, Animals, and Food
- Step 3 : Get the user’s horizontal input
- Step 4 : Move the player left-to-right
- Step 5 : Keep the player inbounds
- Step 6 : Clean up your code and variables

Player Positioning – Step 1: Setup

1. เปิด Unity Hub และสร้างโปรเจคชื่อ "Prototype02"
2. Download Prototype 2 Starter Files
3. From the Project window, open the Prototype 2 scene and delete the SampleScene



Animals, and Food

1. If you want, drag a different material from Course Library > Materials onto the Ground object
2. Drag 1 Human, 3 Animals, and 1 Food object into the Hierarchy
3. Rename the human “Player”, then reposition the animals and food so you can see them
4. Adjust the XYZ scale of the food so you ca easily see it from above

- New Technique: Adjusting Scale
- Warning: Don’t choose people for anything but the player, they don’t have walking animations
- Tip: Remember, dragging objects into the hierarchy puts them at the origin

horizontal input

If we want to move the Player left-to-right, we need a variable tracking the user's input.

1. In your Assets folder, create a "Scripts" folder, and a "PlayerController" script inside
2. **Attach** the script to the Player and open it
3. At the top of PlayerController.cs, declare a new **public float horizontalInput**
4. In **Update()**, set **horizontalInput = Input.GetAxis("Horizontal")**, then test to make sure it works in the inspector

```
public float horizontalInput;

void Update()
{
  horizontalInput = Input.GetAxis("Horizontal");
}
```

to-right

We have to actually use the horizontal input to translate the Player left and right.

1. Declare a new public float speed = 10.0f;
2. **In Update(), Translate the player side-to-side based** on horizontalInput and speed

```
public float horizontalInput;
public float speed = 10.0f;

void Update()
{
  horizontalInput = Input.GetAxis("Horizontal");
  transform.Translate(Vector3.right * horizontalInput * Time.deltaTime * speed);
}
```

inbounds

We have to prevent the player from going off the side of the screen with an if-then statement.

- **Tip:** Move the player in scene view to determine the x positions of the left and right bounds
- **New Concept:** If-then statements
- **New Concept:** Greater than > and Less Than < operators

1. In Update(), write an if-statement checking if the player's left X position is less than a certain value
2. In the if-statement, set the player's position to its current position, but with a fixed X location

```
void Update() {
  if (transform.position.x < -10) {
    transform.position = new Vector3(-10, transform.position.y, transform.position.z);
  }
}
```

and variables

We need to make this work on the right side, too, then clean

1. Repeat this process for the right side of the scre
2. Declare new xRange variable, then replace
   the hardcoded values with them
3. Add comments to your code

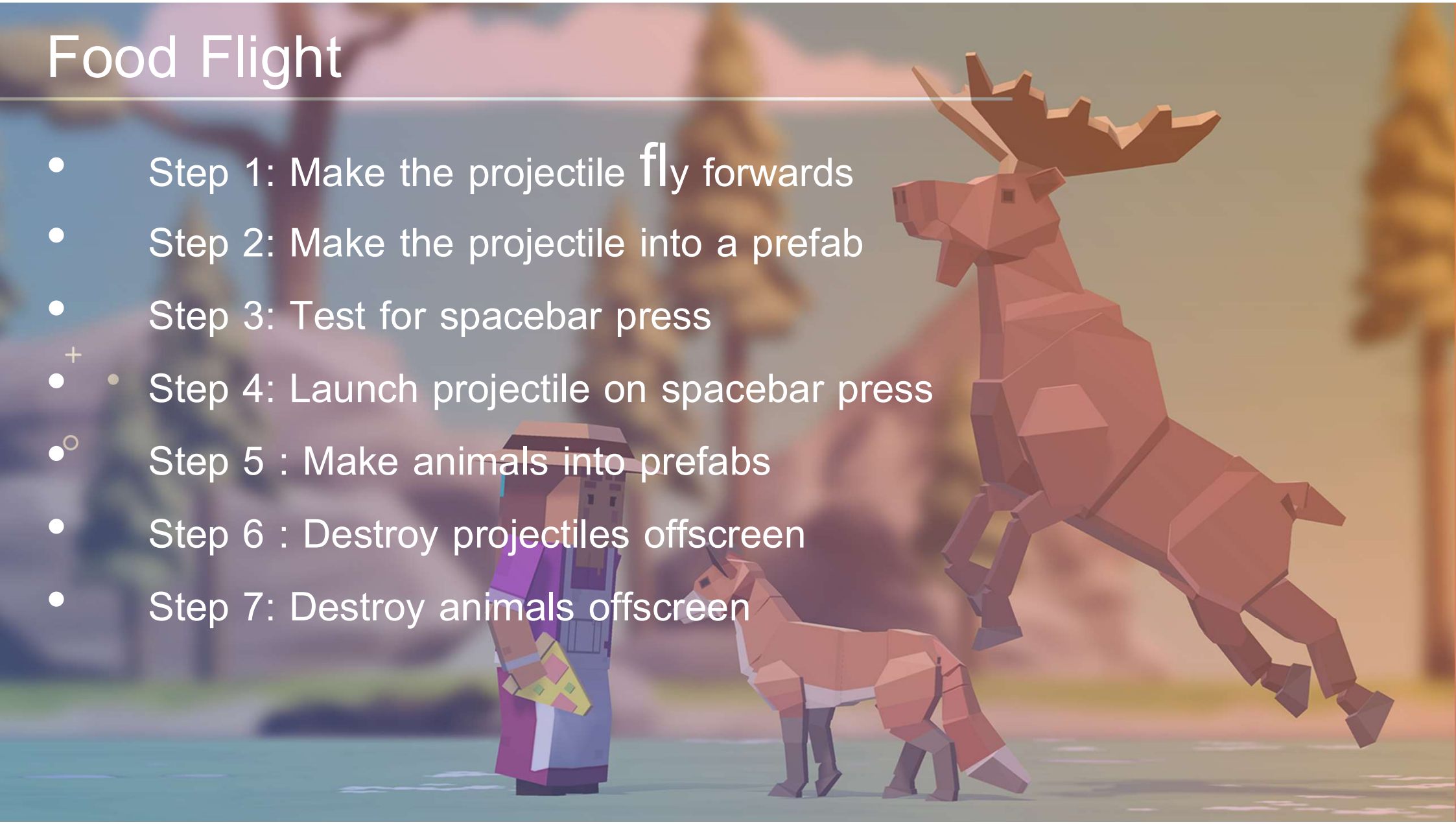
- **Warning:** Whenever you see hardcoded values in the body of your code, try to replace it with a variable
- **Warning:** Watch your greater than / less than signs!

```
public float xRange = 10;

void Update()
{
  // Keep the player in bounds
  if (transform.position.x < ~~-10~~ -xRange)
  {
    transform.position = new Vector3(~~-10~~ -xRange, transform.position.y, transform.position.z);
  }
  if (transform.position.x > xRange)
  {
    transform.position = new Vector3(xRange, transform.position.y, transform.position.z);
  }
}
```


Food Flight - ยิงกระสุน
อาหาร

Food Flight

- Step 1: Make the projectile fly forwards
- Step 2: Make the projectile into a prefab
- Step 3: Test for spacebar press
- Step 4: Launch projectile on spacebar press
- Step 5 : Make animals into prefabs
- Step 6 : Destroy projectiles offscreen
- Step 7: Destroy animals offscreen



forwards

The first thing we must do is give the projectile some forward movement so it can zip across the scene when it's launched by the player.

1. Create a new "MoveForward" script, **attach** it to the food object, then open it
2. Declare a new ***public float speed*** variable;
3. In ***Update()***, add ***transform.Translate(Vector3.forward * Time.deltaTime * speed);***, then **save**
4. In the **Inspector**, set the projectile's **speed** variable, then test

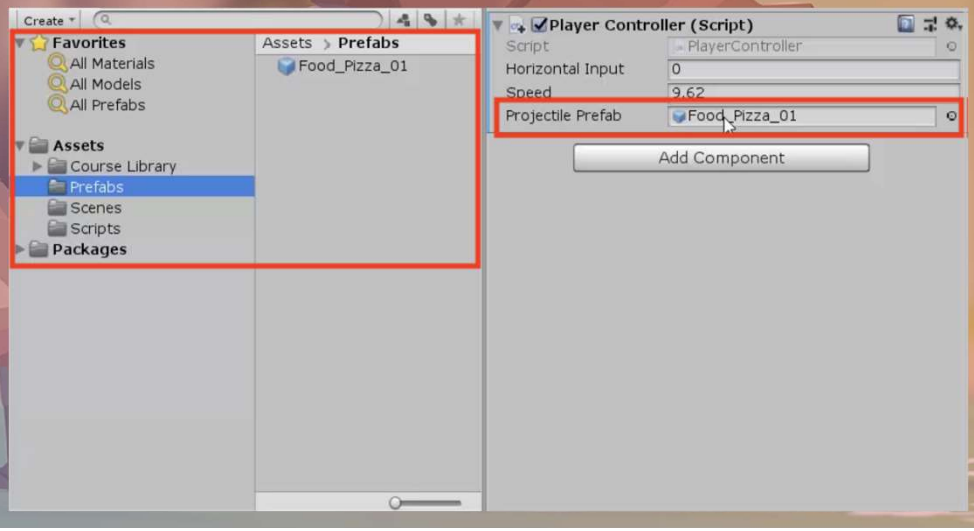
```
public float speed = 40;

void Update() {
  transform.Translate(Vector3.forward * Time.deltaTime * speed);
}
```

prefab

Now that our projectile has the behavior we want, we need to make it into a prefab it so it can be reused anywhere and anytime, with all its behaviors included.

1. Create a new "Prefabs" folder, drag your food into it, and choose **Original Prefab**
2. In PlayerController.cs, declare a new ***public GameObject projectilePrefab;*** variable
3. **Select** the Player in the hierarchy, then **drag** the object from your Prefabs folder onto the new **Projectile Prefab box** in the inspector
4. Try **dragging** the projectile into the scene at runtime to make sure they fly

- **New Concept:** Prefabs
- **New Concept:** Original vs Variant Prefabs
- **Tip:** Notice that this your projectile already has a move script if you drag it in



Food Flight – Step 3: Test for spacebar press

Now that we have a projectile prefab assigned to PlayerController.cs, the player needs a way to launch it with the space bar.

1. In PlayerController.cs, in ***Update()***, add an **if-statement** checking for a spacebar press: ***if (Input.GetKeyDown(KeyCode.Space)) {***
2. Inside the if-statement, add a comment saying that you should ***// Launch a projectile from the player***

- **Tip:** Google a solution. Something like "How to detect key press in Unity"
- **New Functions:** Input.GetKeyDown, GetKeyUp, GetKey
- **New Function:** KeyCode

```
void Update()
{
  if (Input.GetKeyDown(KeyCode.Space))
  {
    // Launch a projectile from the player
  }
}
```

spacebar press

We've created the code that tests if the player presses spacebar, but now we actually need spawn a projectile when that happens

1. Inside the if-statement, use the **Instantiate** method to spawn a projectile at the player's location with the prefab's rotation

- **New Concept:** Instantiation

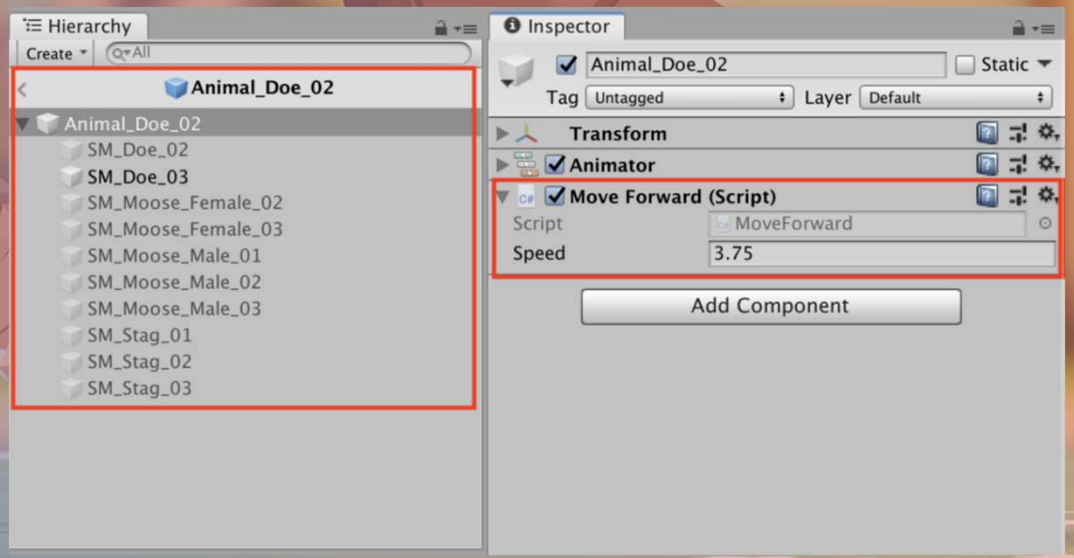
```
if (Input.GetKeyDown(KeyCode.Space))
{
  // Launch a projectile from the player
  Instantiate(projectilePrefab, transform.position, projectilePrefab.transform.rotation);
}
```

Food Flight – Step 5 : Make animals into prefabs

The projectile is now a prefab, but what about the animals? They need to be prefabs too, so they can be instantiated during the game.

1. **Rotate** all animals on the Y axis by **180 degrees** to face down
2. **Select** all three animals in the hierarchy and *Add Component* > ***Move Forward***
3. Edit their **speed values** and **test** to see how it looks
4. Drag all three animals into the **Prefabs folder**, choosing "Original Prefab"
5. **Test** by dragging prefabs into scene view during gameplay

- **Tip:** You can change all animals at once by selecting all them in the hierarchy while holding Cmd/Ctrl
- **Tip:** Adding a Component from inspector is same as dragging it on
- **Warning:** Remember, anything you change while the game is playing will be reverted when you stop it



Food Flight – Step 6 : Destroy projectiles offscreen

Whenever we spawn a projectile, it drifts past the play area into eternity. In order to improve game performance, we need to destroy them when they go out of bounds.

1. Create "DestroyOutOfBounds" script and apply it to the **projectile**
2. Add a new ***private float topBound*** variable and initialize it ***= 30;***
3. Write code to destroy if out of top bounds ***if (transform.position.z > topBound) { Destroy(gameObject); }***
4. In the Inspector **Overrides** drop-down, click **Apply all** to apply it to prefab

- **Warning:** Too many objects in the hierarchy will slow the game
- **Tip:** Google "How to destroy gameobject in Unity"
- **New Function:** Destroy
- **New Technique:** Override prefab

```
private float topBound = 30;

void Update() {
  if (transform.position.z > topBound) {
    Destroy(gameObject);  }}
```

Food Flight – Step 7: Destroy animals offscreen

If we destroy projectiles that go out of bounds, we should probably do the same for animals. We don't want critters getting lost in the endless abyss of Unity Editor...

1. Create a new **private float lowerBound** variable and initialize it = -10;
2. Create **else-if statement** to check if objects are beneath ***lowerBound***:
   else if (transform.position.z > topBound)
3. **Apply** the script to all of the animals, then **Override** the prefabs

- **New Function:** Else-if statement
- **Warning:** Don't make topBound too tight or you'll destroy the animals before they before they can spawn

```
private float topBound = 30;
private float lowerBound = -10;

void Update() {
  if (transform.position.z > topBound)
  {
    Destroy(gameObject);
  } else if (transform.position.z < lowerBound) {
    Destroy(gameObject);
  }
}
```

Next Week

- Feed the Animals (To be continued…)