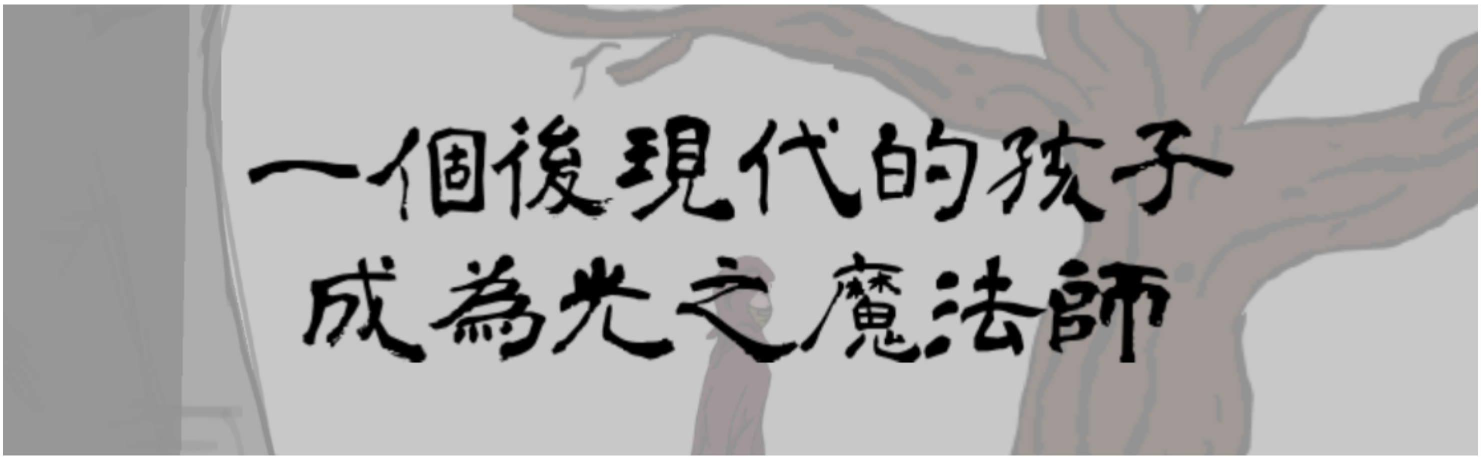
FOOP Final Report



(1) Team member

陳品鈞	b07902113
何政勳	b07902129
葉暘	b07902137

(2) Job Division

陳品鈞

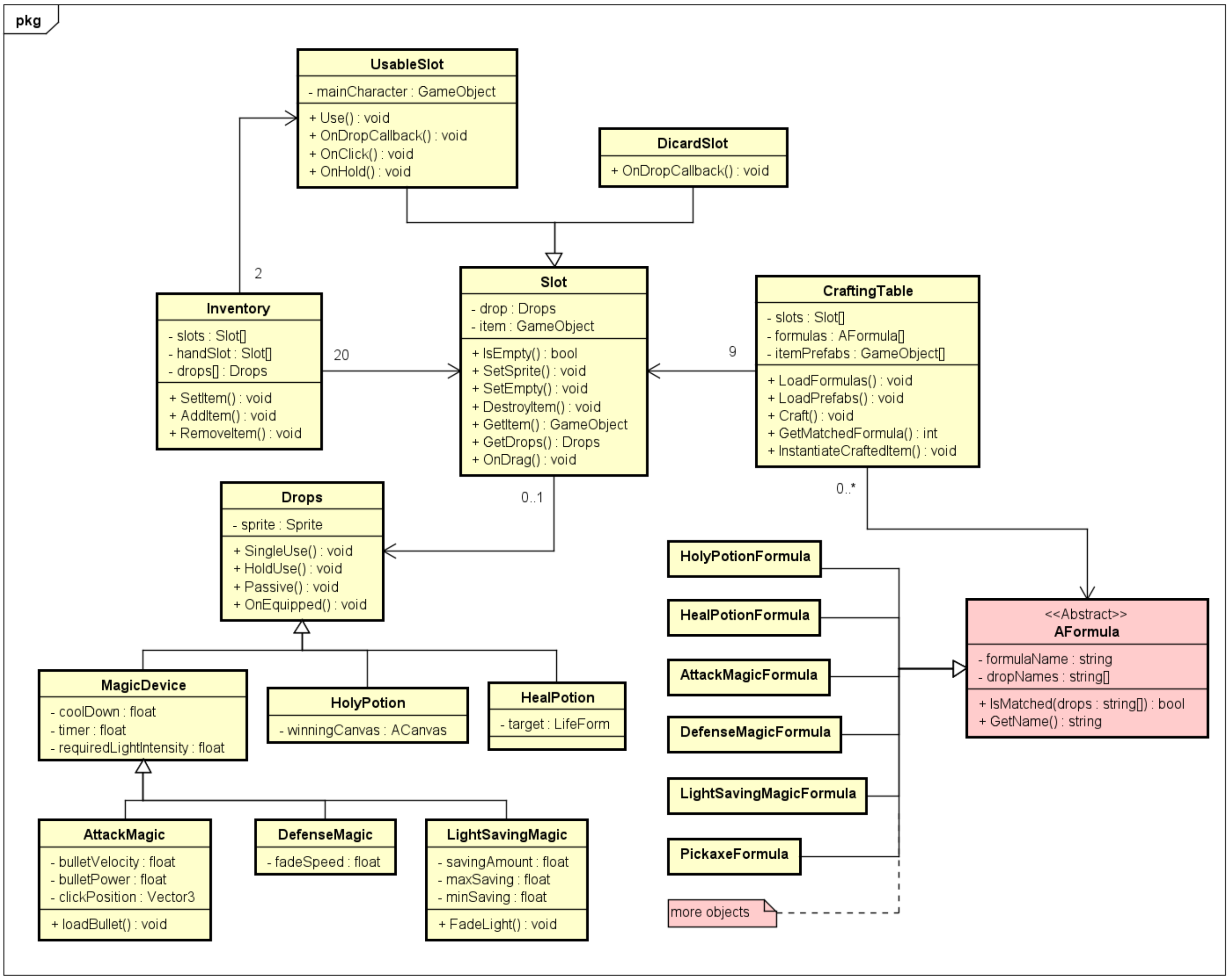
Mainly coded Enemy, Interaction, Inventory, Behavior parts.

何政勳

Mainly coded Crafting, UI, Character Movement, Lifeform parts, Draw the main character animations and some of the images.

葉暘

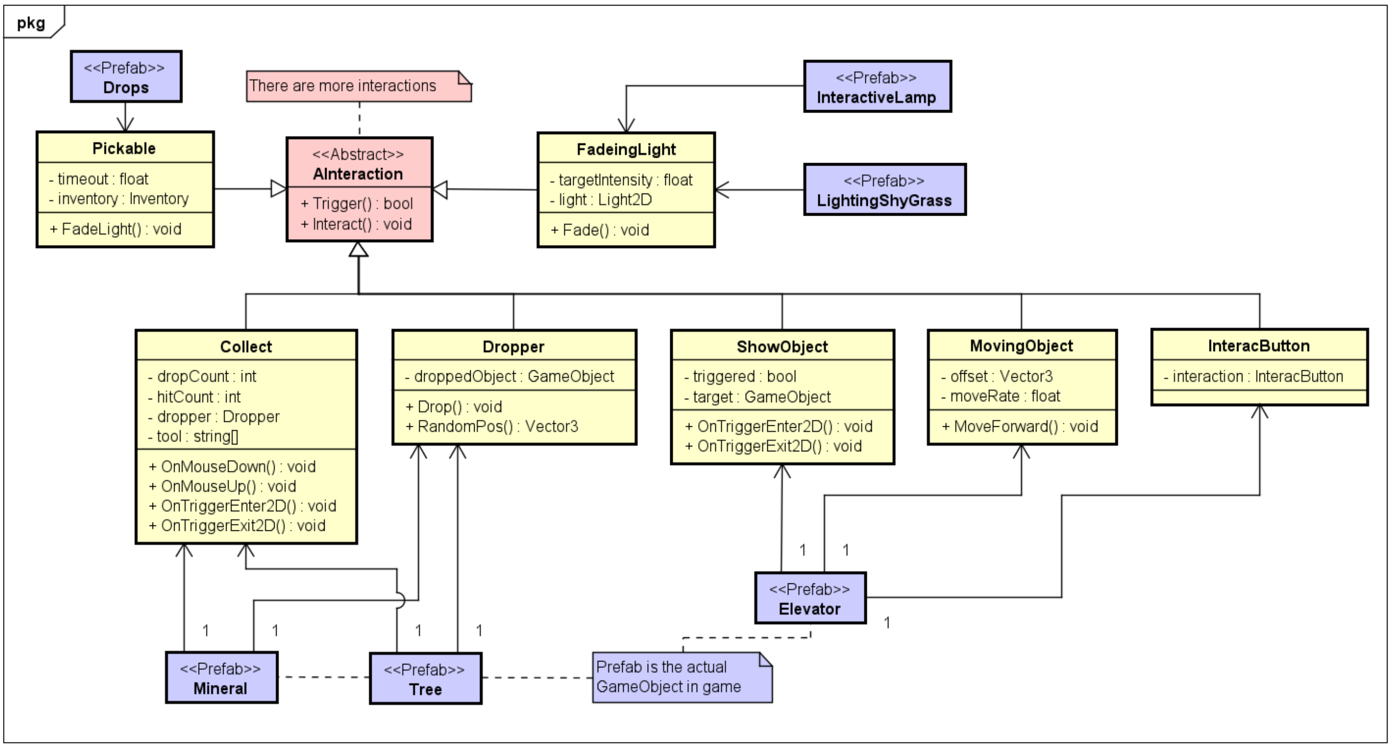
(3) Class Relations



One of the important parts of our games is the crafting system, the figure above is a simplified graph that shows the classes involved in crafting and inventory system.

1. **Drops**
   a. All the craftable objects and items from mining, killing enemies, derived from **Drops**.
   b. All the **Drops** need to implement 3 functions for single pressed ability, hold ability and passive ability, which are called by **UsableSlot**.
   c. **MagicDevice**:
      i. **MagicDevice** is a base class for light sensitive magic devices, which is derived from **Drops**.
2. **AFormula**
   a. **AFormula** is an abstract class for crafting formulas. All craftable items need a corresponding **AFormula**.
   b. **AFormula** is used in CraftingTable.
3. **Slot**
   a. **Slot** is the basic element in **CraftingTable** system and **Inventory** system, which stores a **Drops**

   b. Players can drag around the **Slot**s to organize their inventory or craft new items.
   c. **UsableSlot**
      i. **UsableSlot** handles the hotkey for **Drops** usage.
   d. **DiscardSlot**
      i. **DiscardSlot** deletes the unwanted **Drops**.
4. **Inventory**
   a. **Inventory** saves the **Drops** owned by the players by maintaining a list of **Slot**s.
   b. **Inventory** handles **Drops** insertion and deletion.
5. **CraftingTable**
   a. **CraftingTable** saves all the **AFormula** and the corresponding game objects for crafting.
   b. **CraftingTable** maintains a lists of **Slot**. When the player crafts, **CraftingTable** compares the **Drops** stored in those **Slot**s with all the **AFormula**s.



The second important parts of our game is the interaction between main character and the objects in the game.

1. **Prefab**
   a. **Prefab** is an Unity Object that saves multiple game objects, scripts, components as a large, reusable game object.
2. **AInteraction**
   a. **AInteraction** is the base class for all the character-object interactions.
   b. Interaction() is called if the Triggered() returns true.
   c. **Pickable**
      i. **Pickable** interaction is used in all the **Drops** that players can collect them into their **Inventory**.
   d. **Collect**
      i. **Collect** handles the mining or resource collecting process.
      ii. **Collect** requires a **Dropper**, which will generate the collected resource.
   e. **Dropper**
      i. **Dropper** interaction is used when an object needs to generate and drop a certain game object.

ii. **Dropper** is involved in Collect, enemy loot, etc.

f. **ShowObject**

i. **ShowObject** shows a game object when the main character enters its collider.

g. **MovingObject**

i. **MovingObject** is responsible for moving objects back and forth when it is triggered.

ii. **MovingObject** is used in elevators.

h. InteractiveButton

i. **InteractiveButton** is used to trigger another interaction.

3. Examples

a. Mining Process

i. Mining process consists of two interactions, **Collect** and **Dropper**.

ii. **Collect** counts how many times the player has clicked, if it meets the requirement, then the **Dropper** is called.

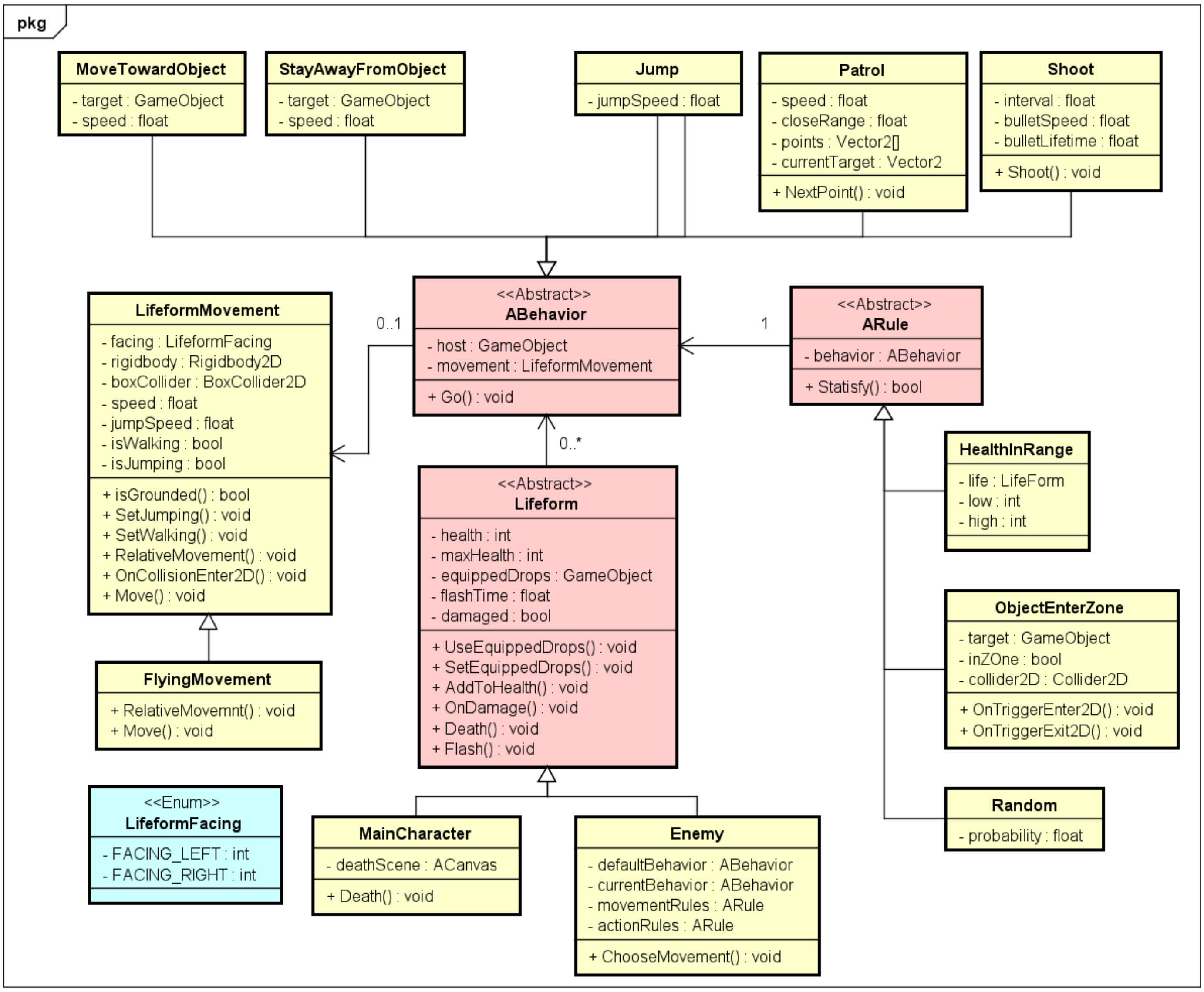
iii. **Dropper** generates and drops the resource.

b. Elevator

i. Elevator consists of three interactions, **ShowObject**, **MovingObject** and **InteractiveButton**.

ii. When the main character enters the elevator, **ShowObject** shows an **InteractiveButton**.

iii. When the **InteractiveButton** is clicked, it triggers the **MovingObject** interaction, allowing the elevator to move.



Creatures are an important aspect of our game, as battling is a huge part of it, besides basic lifeform and their movement. We further create two abstract classes for managing other creatures' behaviors.

1. **Lifeform**
    a. **Lifeform** is the base class of living things, manages taking damage, death and equip items.
    b. **MainCharacter** triggers death scene on death.
    c. **Enemy** moves according to its movementRules, actionRules and defaultMovement. As movements conflict with each other(like **MoveTowards** and **AwayFrom**), enemies will only do one of the behaviors which rule satisfies in movementRule. But all of the behavior in actionRules will be done if satisfied.
        i. **DropEnemy** drops items when dead.
2. **LifeformMovement**
    a. **LifeformMovement** handles the base of lifeform movements.
    b. **FlyableMovement** can move freely in the y direction.
3. **ABehaviors**
    a. **ABehavior** is an abstract class which requires implementation of method go().
    b. **MoveTowards** makes **Lifeform** move towards a specified object.
    c. **AwayFrom** makes **Lifeform** move away from a specified object.

    d. **Jump** makes **Lifeform** jumps.
    e. Patrol makes Lifeform moves between a set of points in order.
    f. Shoot makes Lifeform fire projectiles at the specified target.

4. ARule
    a. ARule is an abstract class that requires method Satisfy() and an attaching behavior.
    b. HealthInRange is satisfied when target health is in the specified interval.
    c. ObjectEnterZone is satisfied when the target enters a specified zone(Collider).
    d. Random satisfies with the specified probability.

Other classes

1. **ACanvas**:
    a. **ACanvas** is the base class for all the UI canvas. It handles the transition animation between two UIs.
2. **PrefabLoader**:
    a. **PrefabLoader** loads prefab from disk.

(4) Advantage of our design.

We break each game element into small classes, so it is easy to recycle different elements and make a lot of GameObjects.

The base class **Drops** defines the interface for transmitting our custom items between different systems. The format for a Magic Device, a loot, a resource are all the same. This allows us to make totally different types of items that can be placed in the inventory and be used when equipped.

Slot base system is also easy to use and extend, which appears in Crafting, Inventory, Hotkey for items (UsableSlot), Discarding unwanted items (DiscardSlots). Slot captures the basic function of dragging items around to manage items.

Enemy movements are also broken into peices. We can assemble different components to create a new behavior enemy. Such as if we mix these existing components:

- CheckHealth
- ObjectEnterZone
- CollideActivator
- AwayForm
- Shoot

We can make a new enemy that shoots projectiles at long distances, flees when health is low, and disappears when it touches fire.

(5) Disadvantage of our design

Since we want to make our design flexible and our classes minimal, every big GameObject contains a lot of components, and each component has a lot of attributes to assign, which feels kind of excessive.
For instance, we make the MoveTowards Class have the free attribute target rather than setting it to our main character for flexibility. It results in we have to assign it for every monster with MoveTowards. We could have just made MoveTowardsPlayer and created extended class if we wanted the flexibility.

(6) Packages and platform

We developed our game on Unity. Scripts were coded in C#.
Other than standard unity packages, we imported TextMeshPro for displaying some text UI.

(7) How to play our game

How to build

1. Clone our project from our repository.
2. Use Unity Hub, select Add project and navigate to src/ directory.
3. Open Demo Scene 1 under src/Assets/Scenes.
4. Press the play button located at the center top.
5. Our game supports touch input, so one can build our game into an Android app following the manual.
6. One can also download our pre-build apk file from here.

In this game

You are a light sorcerer that can use magic items with the presence of light. You had heard about a place that has holy water guarded by monsters. Rumors have that whoever has the holy water can become immortal. Thus starts your journey.

You can collect resources and craft items with them, items are usable when they are placed in the usable slots. Beat monsters in the way and craft out holy potion to win!

How to Attack

As a sorcerer, you do not have physical ways of attack, in compensation, you use bullet magic to do physical damage for you.

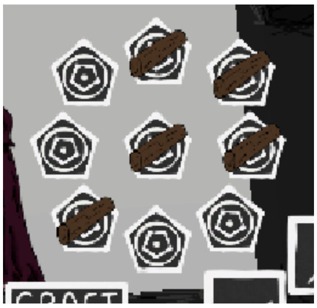
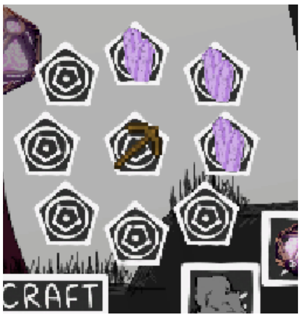
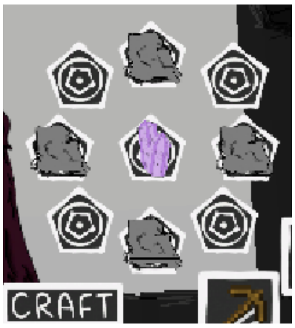
How to Collect

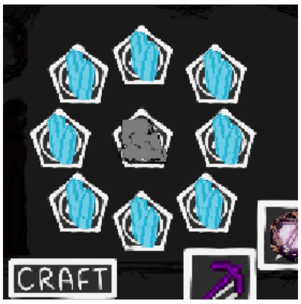
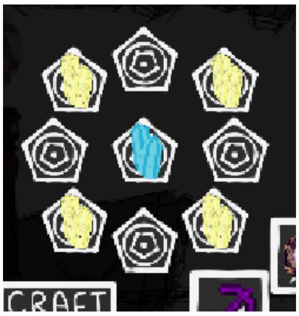
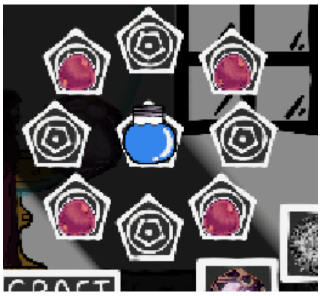
Crystals and trees are collectable, rocks laid in the cave are also collectable. Click on them and if they flash, then it means you are collecting. You need to have tools (Lv1 or 2) in your usable slot to collect crystals (purple(Lv1),yellow and blue(Lv2)) or rock(Lv1).

How to Craft

Like minecraft, if the placing is viable then the crafted item will appear after clicking craft.

The crafting table is as below:

Item name	Item ability	Crafting formula
pickaxe	Tool for mining purple crystal and rock.	
Lv2 pickaxe	Tool for mining blue and yellow crystals.	
Bullet magic	Tap the slot to activate. After activation, click on the screen to shoot. It needs to be exposed under light to shoot bullets. The intensity of light affects the power of bullets.	

Bullet Barrier	It blocks enemy projectiles, great against ghosts.	
Light Saver	Saves up light if exposed in high intensity of light. Automatically emits saved light when active if it enters a dark area.	
Health potion	Heal 300 HP.	
Life potion	Become immortal, victory.