Element Puzzle Box

A puzzle system about interaction, transformation, and elemental logic.

The player, represented as a white box, pushes and utilizes elemental boxes and environmental devices to trigger a chain of reactions. Through strategic movement and planning, the player aims to reach the goal tile and transform into the target elemental state.

This is a 2D puzzle game inspired by Sokoban mechanics.
The player controls a box that can shift between elemental forms, using environmental interactions and elemental properties to solve puzzles.
By pushing themselves and other boxes toward goal points, players explore the world and restore the balance of elements.

DESIGN CONCEPT

Element Puzzle Box is a puzzle about understanding through interaction.
The player, as a neutral white box, explores how elements—Fire, Water, Earth, Ice, and Gold—react and transform under different conditions.
Each mechanic is designed to be self-explanatory through cause and effect: players learn by acting, observing, and adapting rather than reading instructions.

DESIGNER: Min Yang
PERSONAL ROLES: Designer, Artist, Programmer
DEVELOPMENT DURATION: 2025.1-2025.3

GAME NAME:
元素推箱子/Element Puzzle Box
GAME TYPE:
2D Puzzle / Systemic Interaction
GAME PLATFORM:
PC (Unity Prototype)
TARGET PLAYER: 18+
TOOL USED:
Unity, Photoshop, Illustrator, Figma

INSPIRATION

Minimalist Visual Style (Limbo / Inside)

Core Puzzle Inspiration (Sokoban)





Rule Interaction & Dynamic Systems (Baba Is You)





Flow & Environmental Effects (Human: Fall Flat / Portal 2)



Elemental Reaction Logic (The Legend of Zelda: Breath of the Wild)

Implicit Teaching & Spatial Guidance (Monument Valley)

UI DESIGN

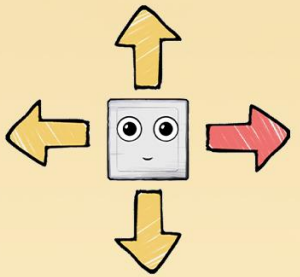
Transform — Press C to choose one of the stored elements and transform the player into that corresponding elemental box.

Press E to select one of the two elements stored inside the player, then choose a direction to release it into an adjacent box.

Press R to rewind and restart the current level.

Esc Press Esc to open the pause or settings menu.

Use Arrow Keys to move the player box and push nearby boxes in the direction you're facing.



Element Output UI

ELEMENT BOX



BLACK



PLAYER



FIRE



WATER



SOLID



GOLD



ICE

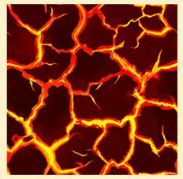


WOOD

ELEMENT FLOOR



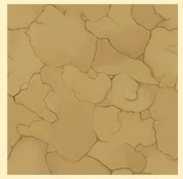
NORMAL



FIRE



WATER



SOLID



GOLD



ICE



PLAYER- NORMAL



PLAYER- FIRE



PLAYER- WATER



PLAYER- SOLID



PLAYER- GOLD



WIND FAN



WATER FLOW



WALL



PLAYER

The player can push any number of boxes, regardless of element type.

The player can absorb and store up to two different elements by stepping on elemental floors.

The player cannot absorb elements from other boxes—only from the ground.

The player can release (output) or transfer one of their stored elements to a nearby box.

Each stored element can only be used once; once released, it disappears from the player's body.



ELEMENT SLOTS

When both slots are filled, no new elements can be absorbed.
After using an element, the player can absorb a new one.

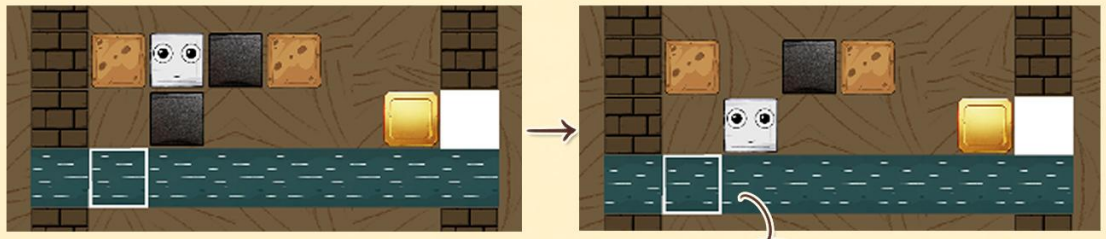
When the player has at least one stored element, they can transform into that element's box by pressing the corresponding key.
This transformation lasts until the end of the level and disables further absorption.

CORE MECHANICS

Water Flow

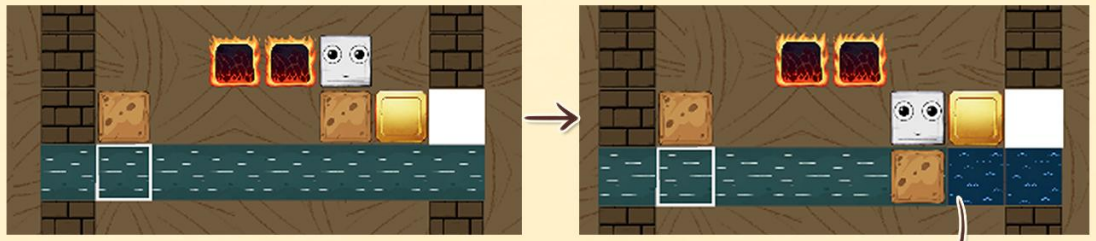
When a box enters a water stream, it is carried away and dissolved — only Solid and Gold boxes can resist the current.

Water flows create dynamic movement in an otherwise static grid, turning spatial puzzles into timing-based reactions.



FLOW CARRIES BOX AWAY

BOX DISSOLVES IN STREAM



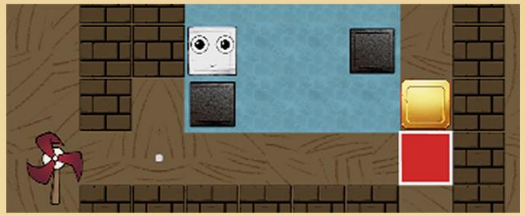
SOLID BLOCKS CURRENT

BLOCKED WATER BECOMES FLOOR ELEMENT

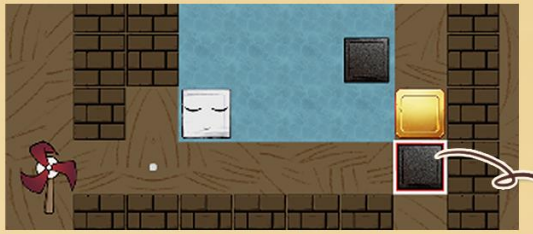
Wind Fan

The wind fan continuously pushes boxes along its direction until they hit an obstacle.

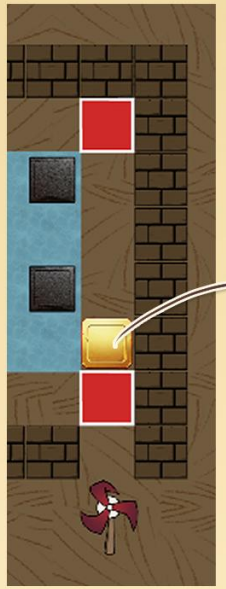
Most boxes are affected by wind, but Gold boxes are heavy enough to resist and can block the airflow.



WIND PUSHES BOX FORWARD



"CONTINUOUS FORCE UNTIL COLLISION"



GOLD RESISTS AIRFLOW

"BLOCKS WIND, PROTECTS BOXES BEHIND"

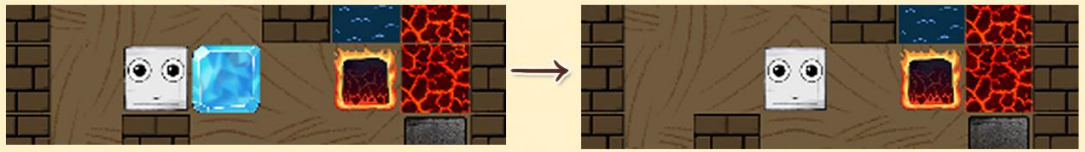
CORE MECHANICS

Fire Box

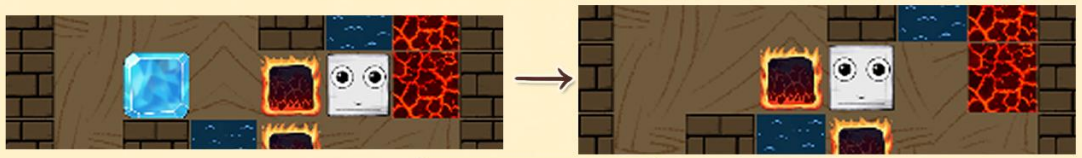
Ice blocks have no friction — once moved, they slide until hitting an obstacle.

When a Fire box stands next to an Ice block, it instantly melts the Ice.

When a sliding Ice block collides into a Fire box, the Fire loses its element and becomes a Black box.



SLIDING ICE EXTINGUISHES FIRE

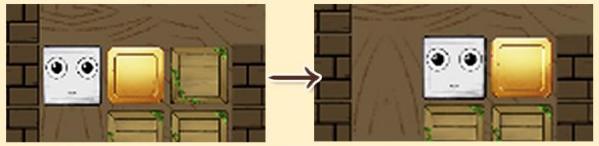


FIRE MELTS ICE

Gold Box

When a Gold box is pushed into a Wood block, the collision destroys the Wood instantly.

Simply being adjacent causes no reaction — impact is required.



IMPACT REACTION

Ice Floor

The Ice Floor has no friction.

When a player or box moves onto it, they continue sliding in the same direction until hitting an obstacle or a non-ice tile.

Adjacent

When elemental boxes touch or collide, they trigger specific reactions based on natural logic.

Some occur instantly when boxes are adjacent, while others require the player to push one box to collide with another.



FIRE AND SOLID ADJACENT
↓
NO REACTION

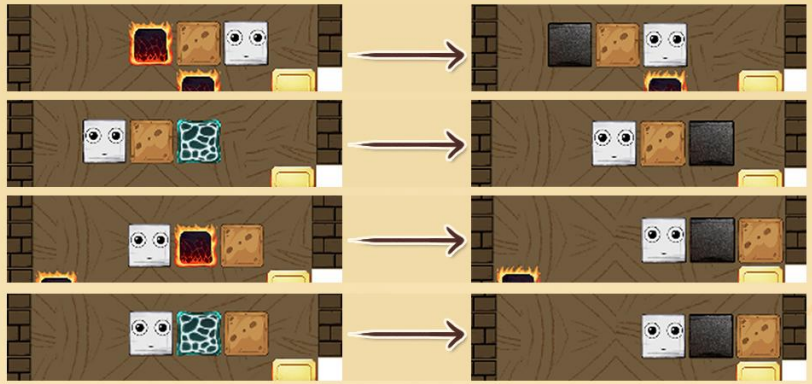


FIRE AND SOLID ADJACENT → NO REACTION



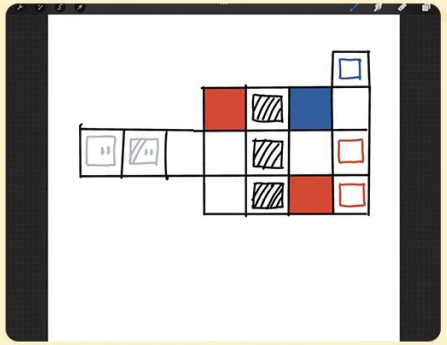
FIRE AND WATER ADJACENT
↓
NO REACTION

Collision

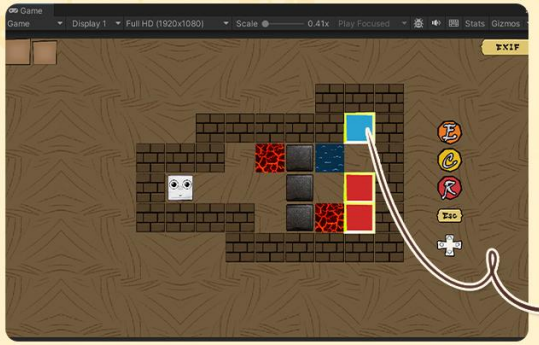


Collision Reaction — When two boxes are already adjacent and the player pushes one to collide, only the active element transforms.
(Fire → Black when pushed into Solid; Water → Black when pushed into Solid.)

LEVEL DESIGN



STAGE 1 SKETCH



STAGE 1

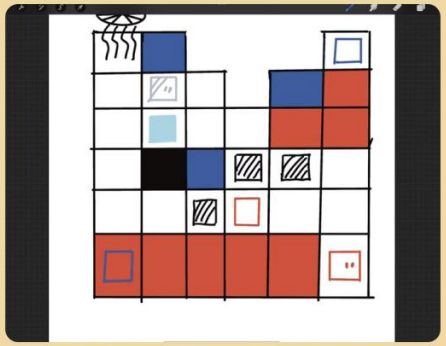
STAGE 1 — INTRODUCTION TO CORE INTERACTION

Teaches movement, pushing, and basic element absorption and output.
The goal is to make the player understand the cause—effect of transferring elements to reach the goal point.

Goal Point — marked by the yellow-white frame.

STAGE 2 — EXPANDING PLAYER UNDERSTANDING

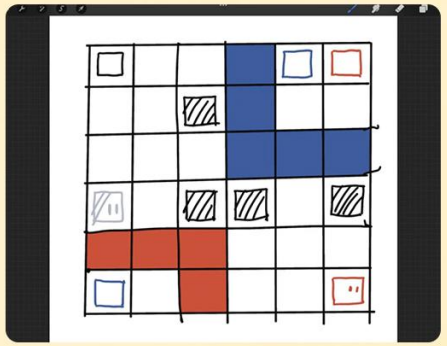
Building upon the foundation of Stage 1, this level deepens the player' s understanding of elemental reactions and their own player mechanics.
A new environmental prop — the Wind Fan — and an Ice Box are introduced, offering more diverse puzzle-solving possibilities.
Through these new mechanics, players learn to think beyond static interactions and explore dynamic, multi-element solutions.reach the goal point.



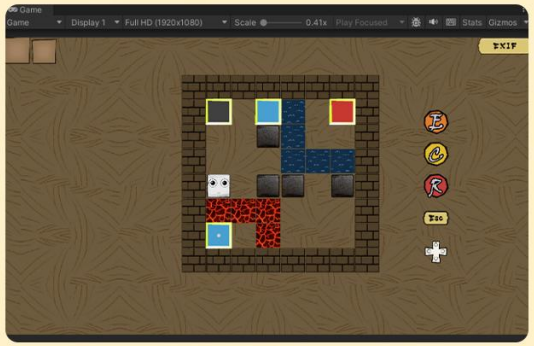
STAGE 2 SKETCH



STAGE 2



STAGE 3 SKETCH

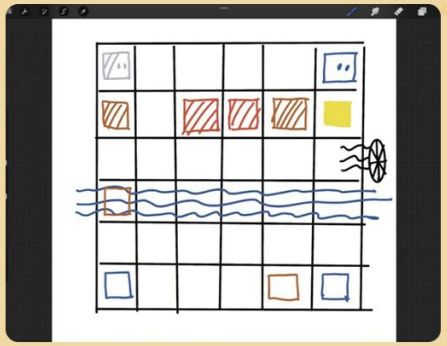


STAGE 3

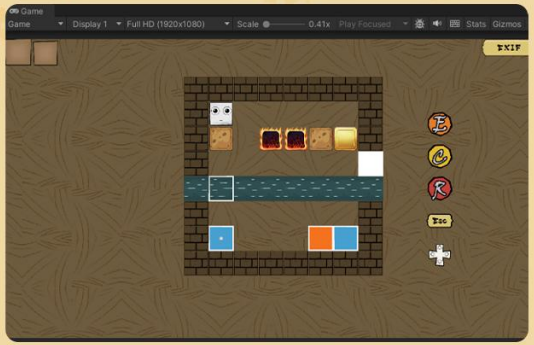
STAGE 3 — MULTI-ELEMENT INTEGRATION

This stage challenges the player to combine multiple elemental reactions within a limited space.
Building on the mechanics introduced in the previous levels, players must now plan sequences and predict interactions between fire, water, and solid blocks.
The puzzle emphasizes order, and spatial logic, marking the transition from basic understanding to strategic thinking.

LEVEL DESIGN



STAGE 4 SKETCH



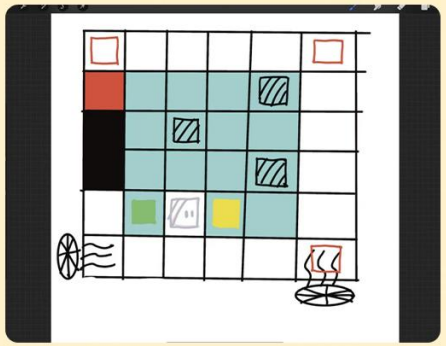
STAGE 4

STAGE 4 — WATER FLOW MECHANIC

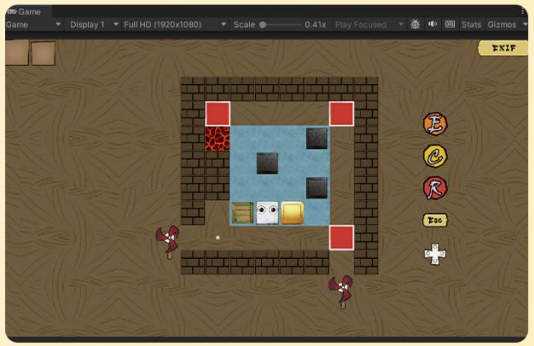
This stage introduces the Water Flow system, where moving streams continuously carry boxes along their direction.
Players must learn how different elements interact with water — most are dissolved, while Earth and Gold boxes can resist the current.
The puzzle focuses on positioning, and obstacle management, transforming static movement into a dynamic rhythm-based challenge.

STAGE 5 — ICE FLOOR MECHANIC

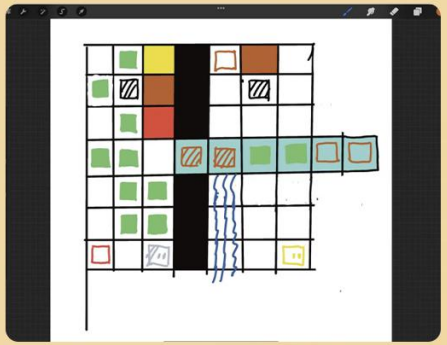
This stage introduces the Ice Floor, a new environmental element that removes friction and changes how movement works.
Once a box or the player starts moving on ice, they will slide continuously until hitting an obstacle.
The level challenges players to plan movement sequences carefully, combining frictionless surfaces with elemental reactions to reach the goal.
It transforms control from precise pushing into momentum-based strategy, deepening the puzzle' s spatial complexity.



STAGE 5 SKETCH



STAGE 5



STAGE 6 SKETCH

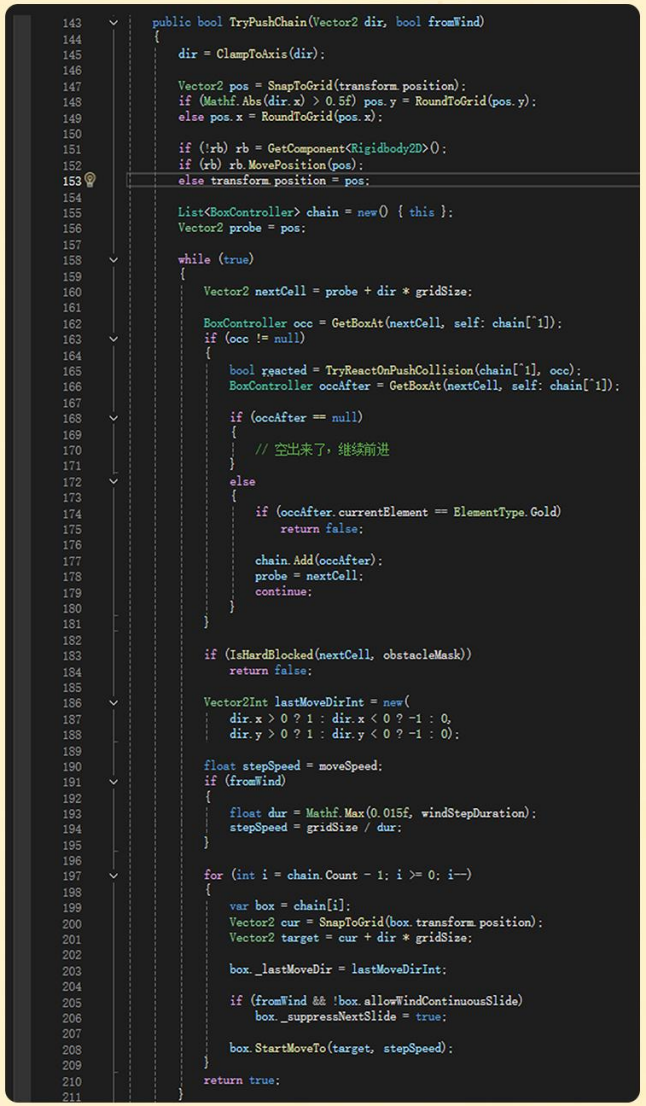


STAGE 6

STAGE 6 — MASTER OF ELEMENTS

The final stage brings together all previously introduced mechanics — fire, water, wind, earth, and gold — into a single integrated puzzle.
Players discover how water flow can be blocked or redirected, and how each element contributes to a larger system.
By combining environmental logic, players demonstrate full mastery of elemental control and problem-solving.

CODING



```
public bool TryPushChain(Vector2 dir, bool fromWind)
{
    dir = ClampToAxis(dir);

    Vector2 pos = SnapToGrid(transform.position);
    if (Mathf.Abs(dir.x) > 0.5f) pos.y = RoundToGrid(pos.y);
    else pos.x = RoundToGrid(pos.x);

    if (!rb) rb = GetComponent<Rigidbody2D>();
    if (rb) rb.MovePosition(pos);
    else transform.position = pos;

    List<BoxController> chain = new() { this };
    Vector2 probe = pos;

    while (true)
    {
        Vector2 nextCell = probe + dir * gridSize;

        BoxController occ = GetBoxAt(nextCell, self: chain[^1]);
        if (occ != null)
        {
            bool reacted = TryReactOnPushCollision(chain[^1], occ);
            BoxController occAfter = GetBoxAt(nextCell, self: chain[^1]);

            if (occAfter == null)
            {
                // 空出来了，继续前进
            }
            else
            {
                if (occAfter.currentElement == ElementType.Gold)
                    return false;

                chain.Add(occAfter);
                probe = nextCell;
                continue;
            }
        }

        if (IsHardBlocked(nextCell, obstacleMask))
            return false;

        Vector2Int lastMoveDirInt = new(
            dir.x > 0 ? 1 : dir.x < 0 ? -1 : 0,
            dir.y > 0 ? 1 : dir.y < 0 ? -1 : 0);

        float stepSpeed = moveSpeed;
        if (fromWind)
        {
            float dur = Mathf.Max(0.015f, windStepDuration);
            stepSpeed = gridSize / dur;
        }

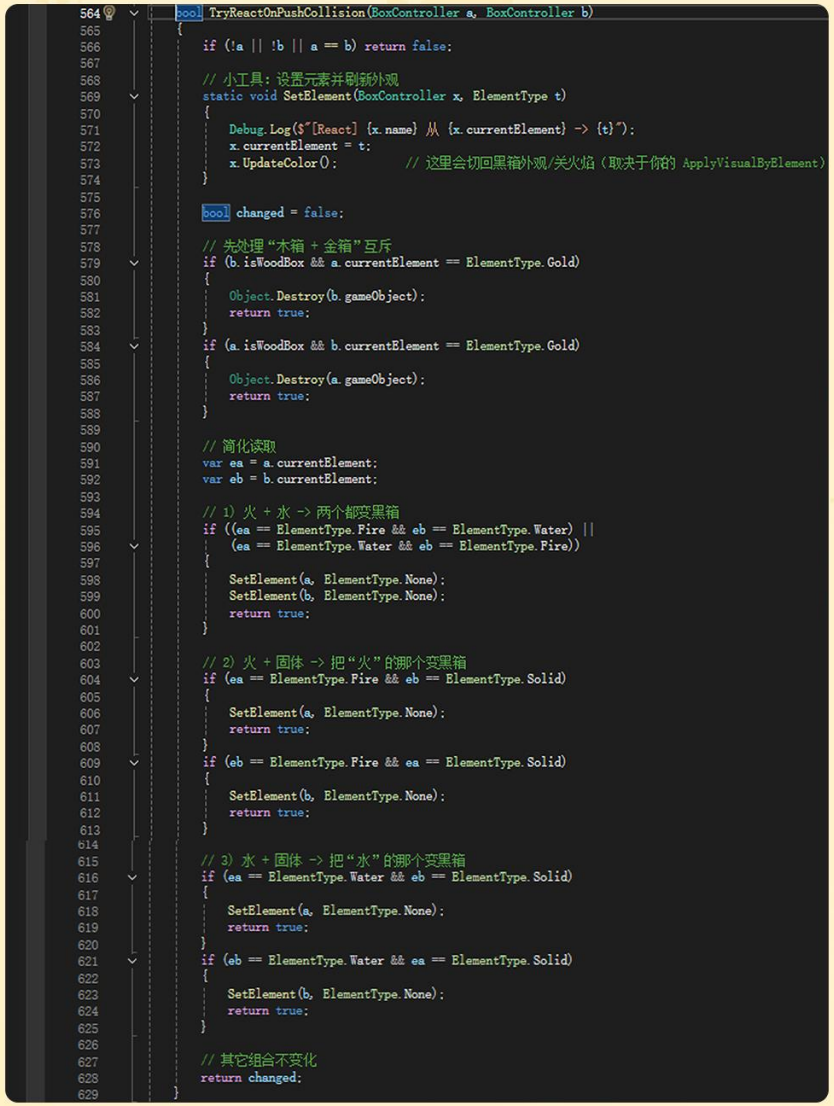
        for (int i = chain.Count - 1; i >= 0; i--)
        {
            var box = chain[i];
            Vector2 cur = SnapToGrid(box.transform.position);
            Vector2 target = cur + dir * gridSize;

            box._lastMoveDir = lastMoveDirInt;

            if (fromWind && !box.allowWindContinuousSlide)
                box._suppressNextSlide = true;

            box.StartMoveTo(target, stepSpeed);
        }
        return true;
    }
```

“BUILDS CHAIN PUSHES AND CHECKS COLLISIONS STEP-BY-STEP.”



```
bool TryReactOnPushCollision(BoxController a, BoxController b)
{
    if (!a || !b || a == b) return false;

    // 小工具：设置元素并刷新外观
    static void SetElement(BoxController x, ElementType t)
    {
        Debug.Log($"[React] {x.name} 从 {x.currentElement} -> {t}");
        x.currentElement = t;
        x.UpdateColor();          // 这里会切回黑箱外观/关火焰（取决于你的 ApplyVisualByElement）
    }

    bool changed = false;

    // 先处理“木箱 + 金箱”互斥
    if (b.isWoodBox && a.currentElement == ElementType.Gold)
    {
        Object.Destroy(b.gameObject);
        return true;
    }
    if (a.isWoodBox && b.currentElement == ElementType.Gold)
    {
        Object.Destroy(a.gameObject);
        return true;
    }

    // 简化读取
    var ea = a.currentElement;
    var eb = b.currentElement;

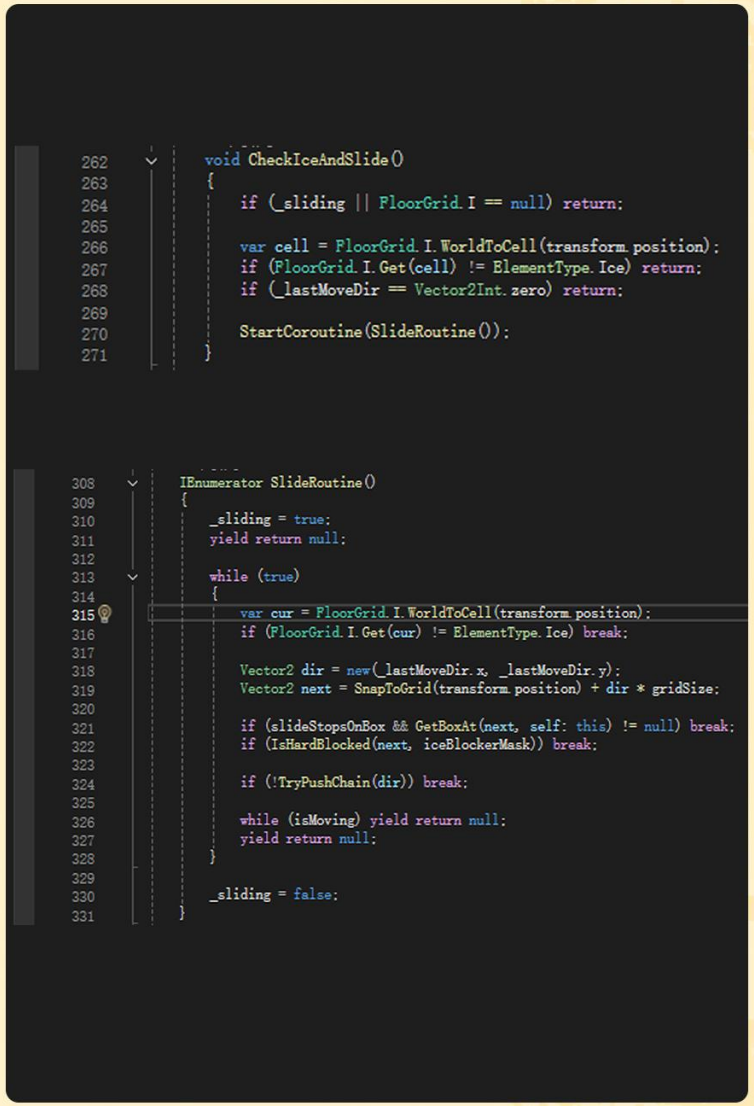
    // 1) 火 + 水 -> 两个都变黑箱
    if ((ea == ElementType.Fire && eb == ElementType.Water) ||
        (ea == ElementType.Water && eb == ElementType.Fire))
    {
        SetElement(a, ElementType.None);
        SetElement(b, ElementType.None);
        return true;
    }

    // 2) 火 + 固体 -> 把“火”的那个变黑箱
    if (ea == ElementType.Fire && eb == ElementType.Solid)
    {
        SetElement(a, ElementType.None);
        return true;
    }
    if (eb == ElementType.Fire && ea == ElementType.Solid)
    {
        SetElement(b, ElementType.None);
        return true;
    }

    // 3) 水 + 固体 -> 把“水”的那个变黑箱
    if (ea == ElementType.Water && eb == ElementType.Solid)
    {
        SetElement(a, ElementType.None);
        return true;
    }
    if (eb == ElementType.Water && ea == ElementType.Solid)
    {
        SetElement(b, ElementType.None);
        return true;
    }

    // 其它组合不变化
    return changed;
}
```

“DEFINES REACTION RULES: FIRE+WATER=BOTH NONE, FIRE/WATER+SOLID→SINGLE CLEAR.”



```
void CheckIceAndSlide()
{
    if (_sliding || FloorGrid.I == null) return;

    var cell = FloorGrid.I.WorldToCell(transform.position);
    if (FloorGrid.I.Get(cell) != ElementType.Ice) return;
    if (_lastMoveDir == Vector2Int.zero) return;

    StartCoroutine(SlideRoutine());
}

IEnumerator SlideRoutine()
{
    _sliding = true;
    yield return null;

    while (true)
    {
        var cur = FloorGrid.I.WorldToCell(transform.position);
        if (FloorGrid.I.Get(cur) != ElementType.Ice) break;

        Vector2 dir = new(_lastMoveDir.x, _lastMoveDir.y);
        Vector2 next = SnapToGrid(transform.position) + dir * gridSize;

        if (slideStopsOnBox && GetBoxAt(next, self: this) != null) break;
        if (IsHardBlocked(next, iceBlockerMask)) break;

        if (!TryPushChain(dir)) break;

        while (isMoving) yield return null;
        yield return null;
    }

    _sliding = false;
}
```

“COROUTINE-DRIVEN ICE SLIDING UNTIL BLOCKED.”

Testing Feedback

In blind playtests, most players intuitively grasped the core mechanics of Element Sokoban—pushing boxes, triggering elemental reactions, and solving puzzles through observation. This validated the effectiveness of my visual cues and level progression.

Some testers, however, struggled to distinguish between different reaction types—such as adjacent contact versus collision-based triggers (e.g., the Fire Box always affects neighboring tiles). This highlighted the need for more explicit visual and audio feedback to clarify system logic. Additionally, feedback suggested that the early levels could introduce mechanics more gradually to improve onboarding and player comprehension.

Self evaluation

Developing Element Sokoban taught me how to translate system logic into readable player experience. I deliberately designed the game without tutorials or text instructions, relying solely on player observation and experimentation to convey rules. Through this process, I realized that puzzle design is not about difficulty—it's about crafting an intuitive learning curve through visual and spatial language.

On the technical side, I deepened my understanding of Unity scripting, prefab organization, and elemental reaction logic, particularly in handling chained triggers and box—floor interactions. As a designer, I learned to think from the player's perspective—to let each level clearly communicate a single idea and reward curiosity through discovery.

Game Review/future plan

In the future, I plan to expand the system by introducing new elemental types and more complex reaction logic.

Each new element should not only add mechanical diversity, but also embody a unique conceptual meaning — reflecting natural relationships such as creation, transformation, or resistance.

For example, Earth and Gold could generate distinctive chain reactions based on their symbolic connection of stability and value.

I also plan to design more advanced levels that emphasize clever spatial composition over pure complexity, allowing players to experience depth through elegance and logic rather than difficulty.