THIẾT KẾ VÀ PHÁT TRIỂN GAME

Bài 6: Clone game kinh điển Tetris

Nội dung

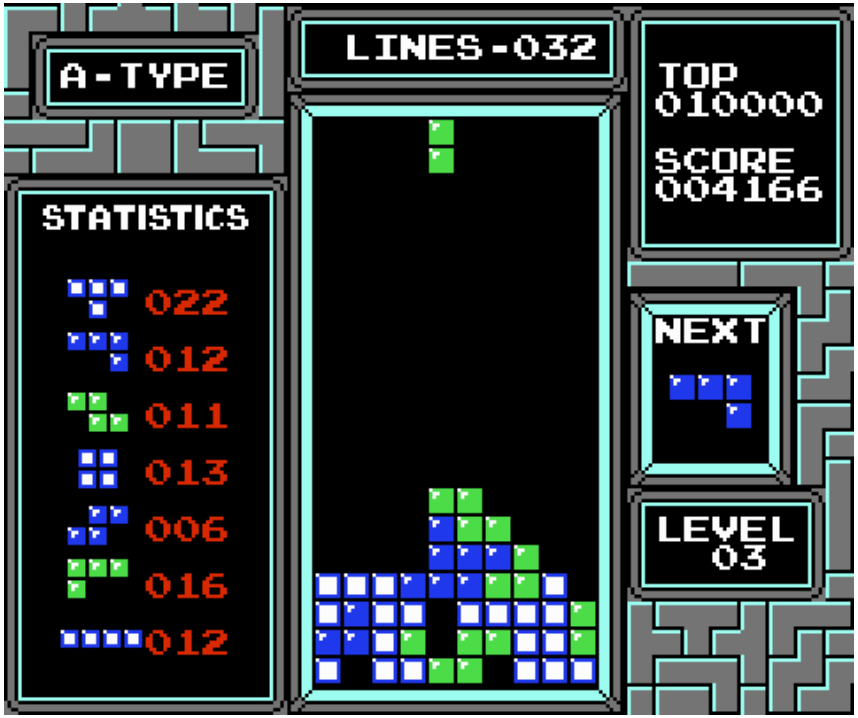
1. Gameplay
2. Chuẩn bị tài nguyên
3. Thiết lập project và các thành phần
4. Cấu trúc dữ liệu chính
5. Xử lý sự kiện
6. Các loại biến cố trong trò chơi
7. Các mở rộng nên xem xét

Phần 1

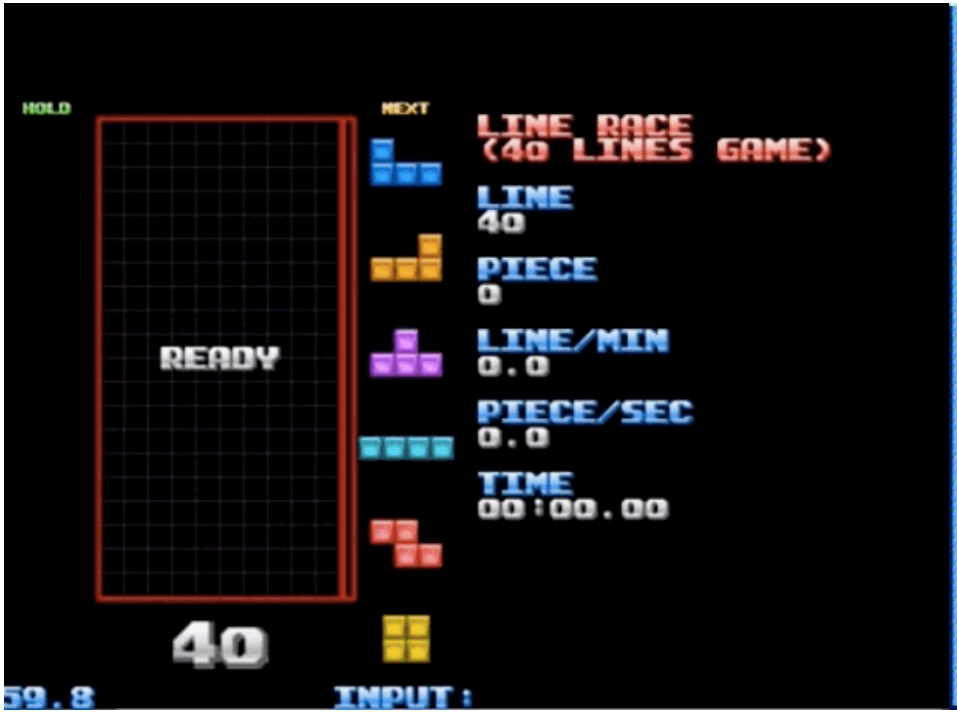
Gameplay

Gameplay

- Game kinh điển, ra đời từ năm 1984
- Sử dụng 7 loại khối cỡ 4, có thể xoay
- Rơi xuống đủ nhanh (tùy level)
- Ăn (và xóa) các dòng đủ
- Mục tiêu: ăn được càng nhiều dòng càng tốt



Gameplay


HOLD
NEXT
LINE RACE
(40 LINES GAME)
LINE
40
PIECE
0
LINE/MIN
0.0
PIECE/SEC
0.0
TIME
00:00.00
READY
40
59.8
INPUT :

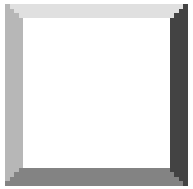

NEXT
2240 SCORE 1445
NEXT
1 LINES 2
7 LEVEL 4
GAME OVER
TETRIS

Phần 2

Chuẩn bị tài nguyên

Chuẩn bị tài nguyên

- Rất đơn giản: sprite duy nhất mô tả 1 block (đây là trong trường hợp đơn giản hóa tối đa bài toán, và không sử dụng các hình ảnh trang trí)

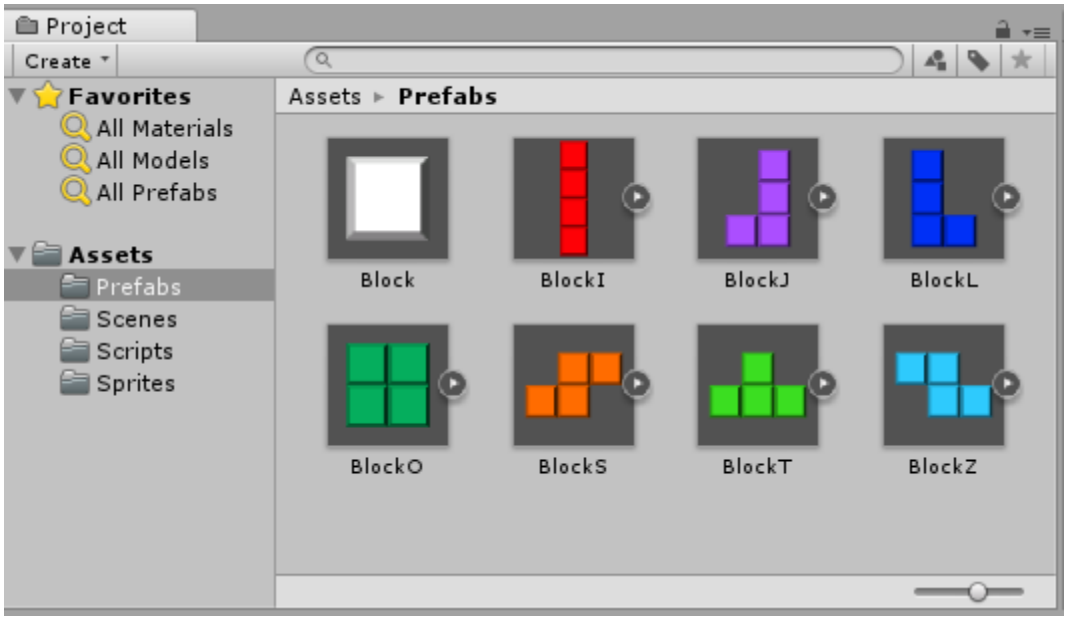


- Nếu clone đúng bản NES có thể phải sử dụng nhiều tài nguyên phức tạp hơn (và code cũng dài hơn)

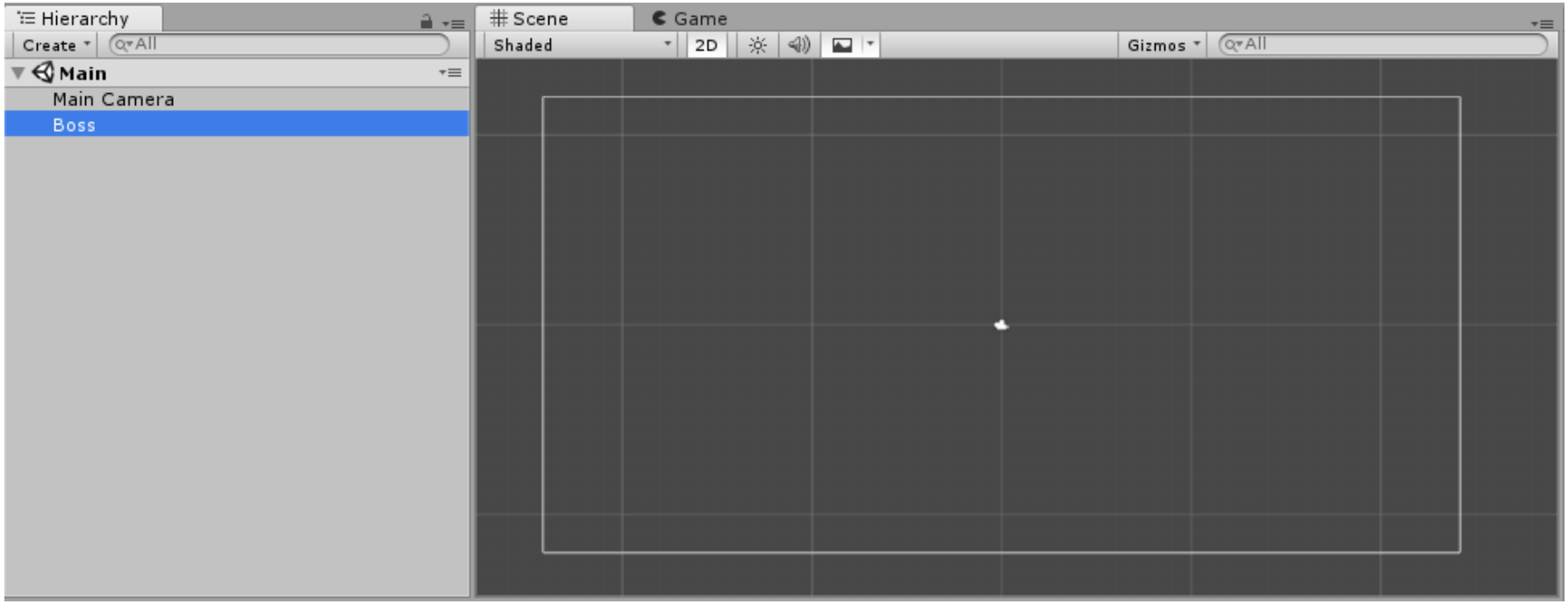
Phần 3

Thiết lập project và các thành phần

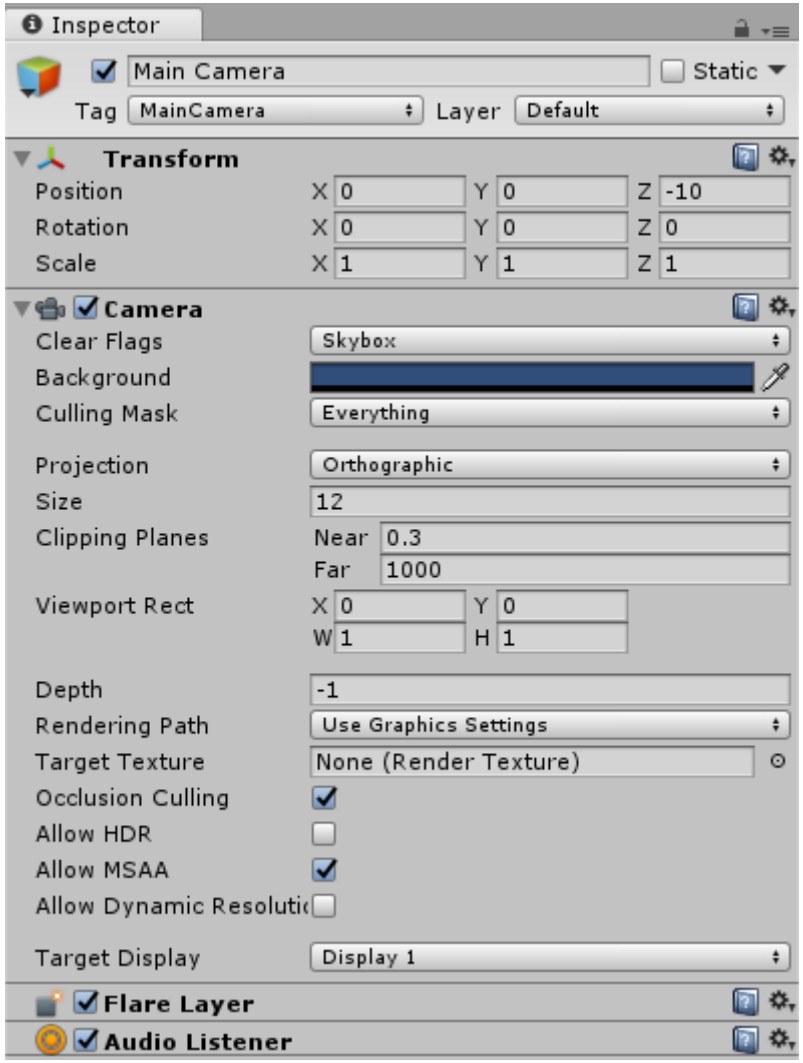
Project và các prefab



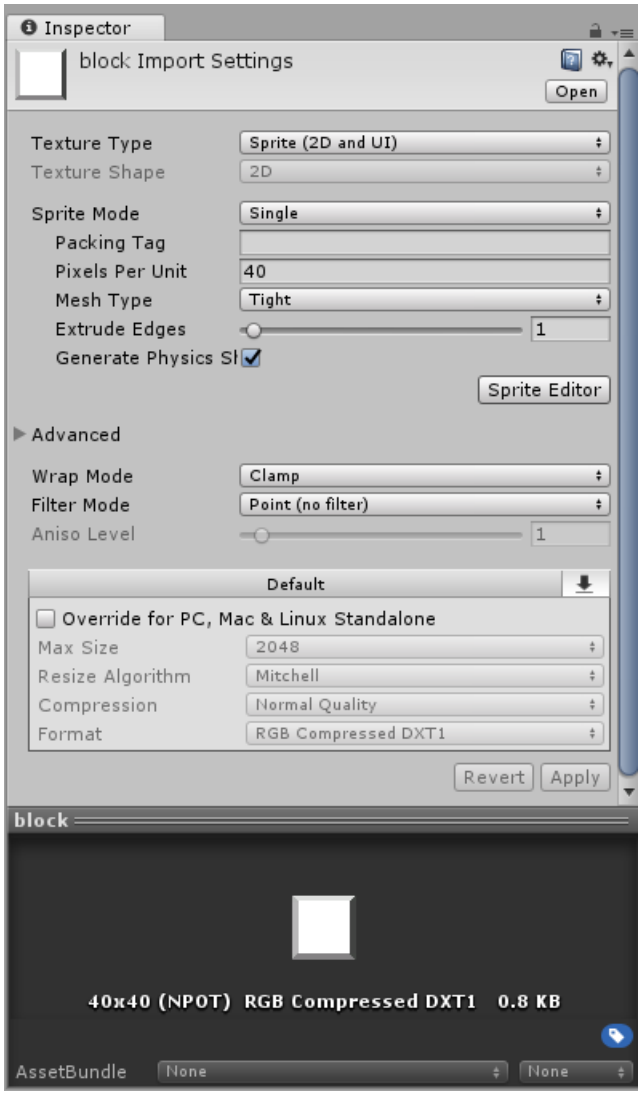
Màn chơi chính



Camera


Inspector
Main Camera
Static
Tag MainCamera
Layer Default
Transform
Position X 0 Y 0 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Camera
Clear Flags Skybox
Background
Culling Mask Everything
Projection Orthographic
Size 12
Clipping Planes Near 0.3 Far 1000
Viewport Rect X 0 Y 0 W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling
Allow HDR
Allow MSAA
Allow Dynamic Resoluti
Target Display Display 1
Flare Layer
Audio Listener

Sprite “block”



Phần 4

Cấu trúc dữ liệu chính

Cấu trúc dữ liệu chính

```
public GameObject whiteBlock;    // khối trắng dùng để xây tường bao
public GameObject[] blocks;      // 7 loại khối
public int maxRow = 20;          // số dòng
public int maxCol = 10;          // số cột
int x0, y0;                      // vị trí trái-dưới (đặt gốc board)
GameObject[,] board = null;      // ma trận các block
GameObject x = null;             // block hiện tại
float lastDown;                  // thời điểm rơi cuối cùng
float speed = 1f;                // tốc độ rơi
// hàm chuyển từ row-col sang vị trí trên scene
Vector3 Board2Pos(int r, int c) {
      return new Vector3 (x0 + c, y0 + r, 0);
}
```

Phần 5

Xử lý sự kiện

Vẽ màn hình chính (1)

```
// khởi tạo màn hình
void initScene() {
    // tính vị trí góc trái-dưới
    x0 = -((maxCol + 2) / 2);
    y0 = -((maxRow + 2) / 2);
    // chuyển board về góc trái-dưới
    transform.position = Board2Pos(0, 0);
    // tạo game board
    board = new GameObject[maxRow + 2, maxCol + 2];
    // xóa toàn bộ board
    for (int i = 0; i < maxRow + 2; i++)
        for (int j = 0; j < maxCol + 2; j++)
            board[i, j] = null;
```

Vẽ màn hình chính (2)

```
    // cot
    for (int i = 0; i < maxRow + 2; i++) {
        // trai
        board[i, 0] = Instantiate(whiteBlock);
        board[i, 0].transform.position = Board2Pos(i, 0);
        // phai
        board[i, maxCol + 1] = Instantiate(whiteBlock);
        board[i, maxCol + 1].transform.position = Board2Pos(i, maxCol + 1);
    }
    // dong
    for (int j = 1; j <= maxCol; j++) {
        // duoi
        board[0, j] = Instantiate(whiteBlock);
        board[0, j].transform.position = Board2Pos(0, j);
        // tren
        board[maxRow + 1, j] = Instantiate(whiteBlock);
        board[maxRow + 1, j].transform.position = Board2Pos(maxRow + 1, j);
    }
}
```

Kiểm tra xung đột

```
bool notOK() {
      foreach (Transform t in x.transform) {
            int col = (int)(t.position.x - x0);
            int row = (int)(t.position.y - y0);
            if (board[row, col] != null)
                  return true;
      }

      return false;
}
```

Dịch trái và dịch phải

```
// dich sang trai
bool left() {
        x.transform.position += Vector3.left;
        if (notOK()) {
                x.transform.position += Vector3.right;
                return false;
        }
        return true;
}
// dich sang phai
bool right() {
        x.transform.position += Vector3.right;
        if (notOK()) {
                x.transform.position += Vector3.left;
                return false;
        }
        return true;
}
```

Hạ xuống

```
// xuong duoi
bool down() {
	x.transform.position += Vector3.down;
	if (notOK()) {
		x.transform.position += Vector3.up;
		return false;
	}
	lastDown = Time.time;
	return true;
}
```

Xoay trái và xoay phải

```
// xoay trai
bool rotateLeft() {
        x.transform.Rotate(0, 0, 90);
        if (notOK()) {
                x.transform.Rotate(0, 0, -90);
                return false;
        }
        return true;
}
// xoay phai
bool rotateRight() {
        x.transform.Rotate(0, 0, -90);
        if (notOK()) {
                x.transform.Rotate(0, 0, 90);
                return false;
        }
        return true;
}
```

Bắt đầu màn chơi

```
// bat dau man choi
void Start () {
	initScene();
	nextBlock();
	lastDown = Time.time;
}
```

Xử lý sự kiện

```
void Update () {
	if (Input.GetKeyDown(KeyCode.LeftArrow)) left();
	if (Input.GetKeyDown(KeyCode.RightArrow)) right();
	if (Input.GetKeyDown(KeyCode.DownArrow)) down();
	if (Input.GetKeyDown(KeyCode.UpArrow)) rotateLeft();
	if (Input.GetKeyDown(KeyCode.End)) rotateRight();
	if (Input.GetKeyDown(KeyCode.Space))
		while (true)
			if (!down()) break;
	if (lastDown + speed < Time.time) {
		if (!down()) nextBlock();
		lastDown = Time.time;
	}
}
```

Phần 6

Các loại biến cố trong trò chơi

Kiểm tra xem row r có full không

```
bool fullRow(int r) {
      for (int j = 1; j <= maxCol; j++)
            if (board[r, j] == null)
                  return false;
      return true;
}
```

Xóa dòng thứ r

```
void clearRow(int r) {
      for (int j = 1; j <= maxCol; j++)
            Destroy(board[r, j]);
      for (int i = r; i < maxRow; i++)
            for (int j = 1; j <= maxCol; j++) {
                  board[i, j] = board[i + 1, j];
                  if (board[i, j] != null)
                  board[i, j].transform.position += Vector3.down;
            }
      for (int j = 1; j <= maxCol; j++)
            board[maxRow, j] = null;
}
```

Duyệt và xóa các dòng bị đầy

```
void clearFullRow() {
      for (int i = 1; i <= maxRow; i++)
            while (fullRow(i))
                  clearRow(i);
}
```

Gắn khối vừa hạ xuống vào board

```
void landing() {
        while (x.transform.childCount > 0) {
                // lay block con
                Transform c = x.transform.GetChild(0);
                // dat block vao board
                int col = (int)(c.position.x - x0);
                int row = (int)(c.position.y - y0);
                board[row, col] = c.gameObject;
                // xoa block khoi khoi hien tai
                c.SetParent(transform);
        }
        // huy khoi hien tai
        Destroy(x);
        x = null;
        clearFullRow();
}
```

Tạo khối tiếp theo

```
// khoi tiep theo
void nextBlock() {
	if (x != null)
		landing();
	// tao ngau nhien mot khoi
	x = Instantiate(blocks [Random.Range(0, 6)], transform);
	x.transform.position = Board2Pos(maxRow - 2, maxCol / 2);
	// Game over
	if (notOK()) {
		???
	}
}
```

Phần 7

Các mở rộng nên xem xét

Các mở rộng nên xem xét

- Làm phong phú thêm các loại block: có thể xét tới các block cỡ 5 hoặc cao hơn
- Trạng thái ban đầu phức tạp (có sẵn nhiều block từ trước)
- Các đồ vật thay đổi trạng thái người chơi
- Chế độ thi đấu: 2 người, qua mạng