```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;   /* need this for the List<> */

public class Main : MonoBehaviour {
    public GameObject cubePrefab_;
    public GameObject spherePrefab_;

    private Ball ball_;
    private List<Block> block_list_;

    // Use this for initialization
    void Start() {
        /* setup camera */
        Camera.main.transform.position = new Vector3(0, 15, 0);
        Camera.main.transform.eulerAngles = new Vector3(90, 0, 0);

        /* setup lighting */
        {
            GameObject go = new GameObject("Light");
            Light light = go.AddComponent<Light>();
            light.transform.position = new Vector3(0, 5, 0);
        }

        /* create walls */
        {
            GameObject wall_left_go = Instantiate(cubePrefab_) as GameObject;
            wall_left_go.transform.localScale = new Vector3(1, 1, 10);
            wall_left_go.transform.position = new Vector3(5, 0, 0);
            GameObject wall_right_go = Instantiate(cubePrefab_) as GameObject;
            wall_right_go.transform.localScale = new Vector3(1, 1, 10);
            wall_right_go.transform.position = new Vector3(-5, 0, 0);
            GameObject wall_top_go = Instantiate(cubePrefab_) as GameObject;
            wall_top_go.transform.localScale = new Vector3(10, 1, 1);
            wall_top_go.transform.position = new Vector3(0, 0, 5);
        }

        /* create paddle */
        {
            GameObject go = Instantiate(cubePrefab_) as GameObject;
            go.transform.localScale = new Vector3(1, 1, 0.5f);
            go.transform.position = new Vector3(0, 0, -5);
            go.AddComponent<Paddle>();
        }

        /* create blocks */
        {
            block_list_ = new List<Block>();
            foreach (var x in new float[]{ -3.5f, -1.5f, 0.5f, 2.5f, }) {
                foreach (var z in new float[]{ 3.9f, 2.9f, }) {
                    GameObject go = Instantiate(cubePrefab_) as GameObject;
                    go.transform.localScale = new Vector3(1.9f, 1f, 0.9f);
                    go.transform.position = new Vector3(x, 0f, z);
                    Block block = go.AddComponent<Block>();
                    block_list_.Add(block);
                }
            }
        }

        /* create ball */
        {
            GameObject go = Instantiate(spherePrefab_) as GameObject;
            Rigidbody rigidbody = go.AddComponent<Rigidbody>();
            /* gravity is not necessary */
            rigidbody.useGravity = false;
            /* add Ball script defined below */
            ball_ = go.AddComponent<Ball>();
            /* setup the first velocity */
            ball_.setVelocity(new Vector3(6f, 0f, 6f));
        }
    }

    // Update is called once per frame
    void Update() {
        block_list_.RemoveAll(s => s == null);
        if (block_list_.Count == 0) {
            ball_.setVelocity(Vector3.zero);
        }
    }
}

public class Ball : MonoBehaviour {
    private Vector3 velocity_;
    public void setVelocity(Vector3 velocity) { velocity_ = velocity; }
    void OnCollisionEnter(Collision collision) {
        if (collision.contacts.Length > 0) {    /* hit for wall or paddle */
            /* calculate the reflecting vector */
            Vector3 p = collision.contacts[0].point;
            Vector3 n = transform.position - p;
            n.Normalize();
            float v = -2 * Vector3.Dot(velocity_, n);
            var r = n * v;
            velocity_ += r; /* change the velocity vector */
        }
    }
    void Update() {
        transform.position += velocity_ * Time.deltaTime;
    }
}

public class Block : MonoBehaviour {
    void OnCollisionEnter(Collision collision) {
        Destroy(gameObject);
    }
}

public class Paddle : MonoBehaviour {
    private Vector3 mouse_position_;

    void Start() {
        /* remember the first position of mouse */
        mouse_position_ = Input.mousePosition;
    }

    void Update() {
        /* the latest position of mouse */
        Vector3 new_mouse_position = Input.mousePosition;
        /* the amount of movement during the last frame */
        Vector3 mouse_delta = new_mouse_position - mouse_position_;
        /* update the paddle position for the next frame */
        mouse_position_ = new_mouse_position;
        /* sensitivity for the paddle */
        float x = transform.position.x + mouse_delta.x * 0.01f;
        /* limit for the paddle */
        x = Mathf.Clamp(x, -5, 5);
        /* update paddle */
        transform.position = new Vector3(x, 0, -5);
    }
}
```