```
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    const int MinLane = -2;
    const int MaxLane = 2;
    const float LaneWidth = 6.0f;
    const float StunDuration = 0.5f;
    float recoverTime = 0.0f;

    public int life = 10;

    CharacterController controller;

    Vector3 moveDirection = Vector3.zero;
    int targetLane;

    public float gravity = 9.81f; //重力

    public float speedZ = 10; //前進方向のスピードの上限値
    public float accelerationZ = 8; //加速度

    public float speedX = 10; //横方向に移動するときのスピード

    public float speedJump = 10; //ジャンプスピード

    public GameObject body;

    public GameObject boms;

    void Start()
    {
        変数 controller に CharacterController コンポーネントを取得
    }

    void Update()
    {
        //ゲームステータスがplayingの時のみ左右に動かせる
        if (GameManager.gameState == GameState.playing)
        {
            if (Input.GetKeyDown(KeyCode.LeftArrow) || Input.GetKeyDown(KeyCode.A)) MoveToLeft();
            if (Input.GetKeyDown(KeyCode.RightArrow) || Input.GetKeyDown(KeyCode.D)) MoveToRight();
            もしもジャンプボタンがおされたらジャンプメソッド
        }

        //もしスタン中かLifeが0なら動きを止める
        if (IsStun())
        {
            moveDirection.x = 0;
            moveDirection.z = 0;
            //復活までの時間をカウント
            recoverTime -= Time.deltaTime;

            //点滅処理
            Blinking();
        }
```

```
    else
    {
        //徐々に加速しZ方向に常に前進させる
        float acceleratedZ = moveDirection.z + (accelerationZ * Time.deltaTime);
        moveDirection.z = Mathf.Clamp(acceleratedZ, 0, speedZ);

        //X方向は目標のポジションまでの差分の割合で速度を計算
        float ratioX = (targetLane * LaneWidth - transform.position.x) / LaneWidth;
        moveDirection.x = ratioX * speedX;
    }

    //重力分の力をフレーム追加
    moveDirection.y -= gravity * Time.deltaTime;

    //移動実行
    Vector3 globalDirection = transform.TransformDirection(moveDirection);
    controller.Move(globalDirection * Time.deltaTime);

    //移動後接地してたらY方向の速度はリセットする
    if (controller.isGrounded) moveDirection.y = 0;

}

//左のレーンに移動を開始
public void MoveToLeft()
{
    もしもスタン中であれば何もせず終了（一行で）
    if (controller.isGrounded && targetLane > MinLane)
        targetLane--;
}

//⇒のレーンに移動を開始
public void MoveToRight()
{
    もしもスタン中であれば何もせず終了（一行で）
    if (controller.isGrounded && targetLane < MaxLane)
        targetLane++;
}

//ジャンプ
public void Jump()
{
    もしもスタン中であれば何もせず終了（一行で）
    //地面に接触していればY方向の力を設定
    if (controller.isGrounded) moveDirection.y = speedJump;
}
```

```
    //体力をリターン
    public int Life()
    {
        return life;
    }

    //スタン中かチェック
    bool IsStun()
    {
        //recoverTimeが作動中かLifeが0になった場合はStunフラグがON
        bool stun = recoverTime > 0.0f || life <= 0;
        //Stunフラグが OFFの場合はボディを確実に表示
        if (!stun) body.SetActive(true);
        //Stunフラグをリターン
        return stun;
    }

    //接触判定
    private void OnControllerColliderHit(ControllerColliderHit hit)
    {
        もしもスタン中であれば何もせず終了（一行で）

        //ぶつかった相手がEnemyなら
        if (hit.gameObject.CompareTag("Enemy"))
        {
            //体力をマイナス
            life--;

            if (life <= 0)
            {
                GameManager.gameState = GameState.gameover;
                Instantiate(boms, transform.position, Quaternion.identity); //爆発エフェクトの発生
                Destroy(gameObject, 0.5f); //少し時間差で自分を消滅
            }
            //recoverTimeの時間を設定
            recoverTime = StunDuration;
            //接触したEnemyを削除
            Destroy(hit.gameObject);
        }
    }

    //点滅処理
    void Blinking()
    {
        //その時のゲーム進行時間で正か負かの値を算出
        float val = Mathf.Sin(Time.time * 50);
        //正の周期なら表示
        if(val >= 0) body.SetActive(true);
        //負の周期なら非表示
        else body.SetActive(false);
    }

}
```