Company Website :www.dinzylabs.com

Contact : admin@dinzylabs.com

Bubble Bird Starter Kit

Bubble Bird Starter Kit is an easiest way of making bubble shooter kind of game. Throw bird of and match colour to pop and drop the birds. Make as many levels by simple tweaking parameters like

1) Total Number of Rows

2) Initial Number of Rows.

3) Row Adding Speed

4) Different pattern types

Features:

- 6 Different Playing Objects
- 7 different in built patterns.
- Colourful graphics and full sound assets
- Create different levels of your own

Important Classes

- LevelManager.cs
- PlayingObject.cs
- PlayingObjectGeneration.cs
- PlayingObjectManager.cs
- Striker.cs
- StrikerManager.cs

Level Manager.cs

Create different levels by tweaking the following static variables :

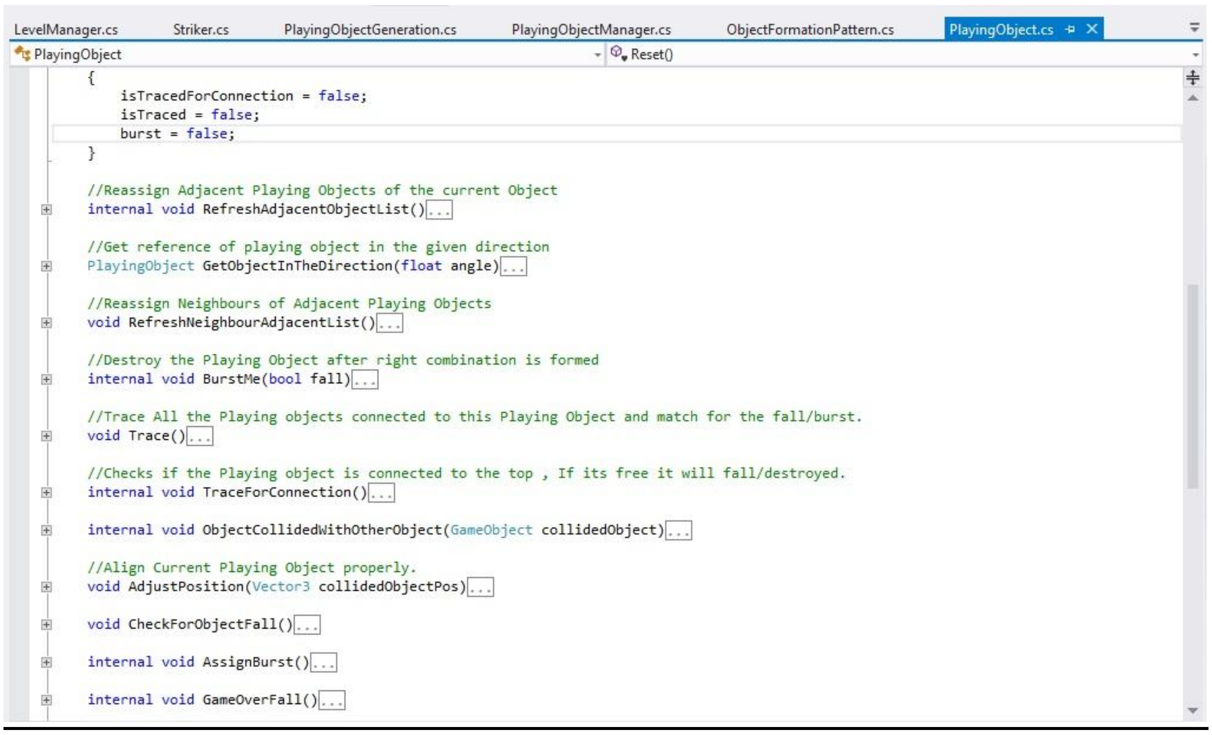
```
PatternTypepatternType;

inttotalNoOfRows;

intminimumNumberOfRow;

introwAddingInterval;
```

PlayingObject.cs

This script is attached to the Playing Object item. It holds the referent of all 6 neighbour playing objects.

Important Functions:

```
RefreshAdjacentObjectList() : Reassign Adjacent Playing Objects of the current Object

GetObjectInTheDirection() : Get reference of playing object in the given direction

RefreshNeighbourAdjacentList(): Reassign Neighbours of Adjacent Playing Objects

Trace() : Trace All the Playing objects connected to this Playing Object and match for
the fall/burst.
```



```
        {
            isTracedForConnection = false;
            isTraced = false;
            burst = false;
        }

        //Reassign Adjacent Playing Objects of the current Object
        internal void RefreshAdjacentObjectList()...

        //Get reference of playing object in the given direction
        PlayingObject GetObjectInTheDirection(float angle)...

        //Reassign Neighbours of Adjacent Playing Objects
        void RefreshNeighbourAdjacentList()...

        //Destroy the Playing Object after right combination is formed
        internal void BurstMe(bool fall)...

        //Trace All the Playing objects connected to this Playing Object and match for the fall/burst.
        void Trace()...

        //Checks if the Playing object is connected to the top , If its free it will fall/destroyed.
        internal void TraceForConnection()...

        internal void ObjectCollidedWithOtherObject(GameObject collidedObject)...

        //Align Current Playing Object properly.
        void AdjustPosition(Vector3 collidedObjectPos)...

        void CheckForObjectFall()...

        internal void AssignBurst()...

        internal void GameOverFall()...
```

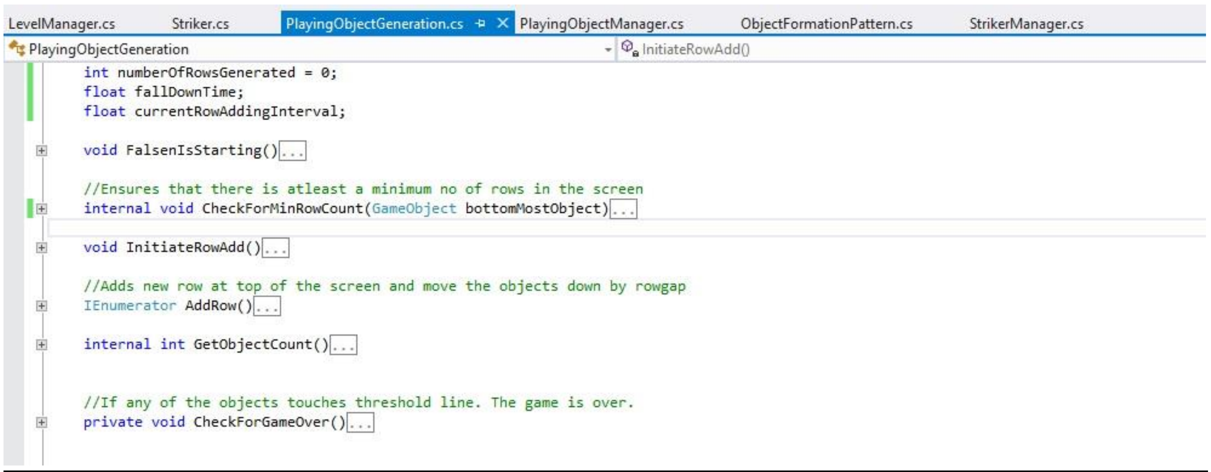
PlayingObjectGeneration.cs

This script is attached to Playing Object Generation GameObject. This is responsible to create new rows in the screen.

Important Functions:

`AddRow()` : Adds new row at top of the screen and move the objects down by row gap.

`CheckForMinRowCount()` : Ensures that there is atleast a minimum no of rows in the screen.



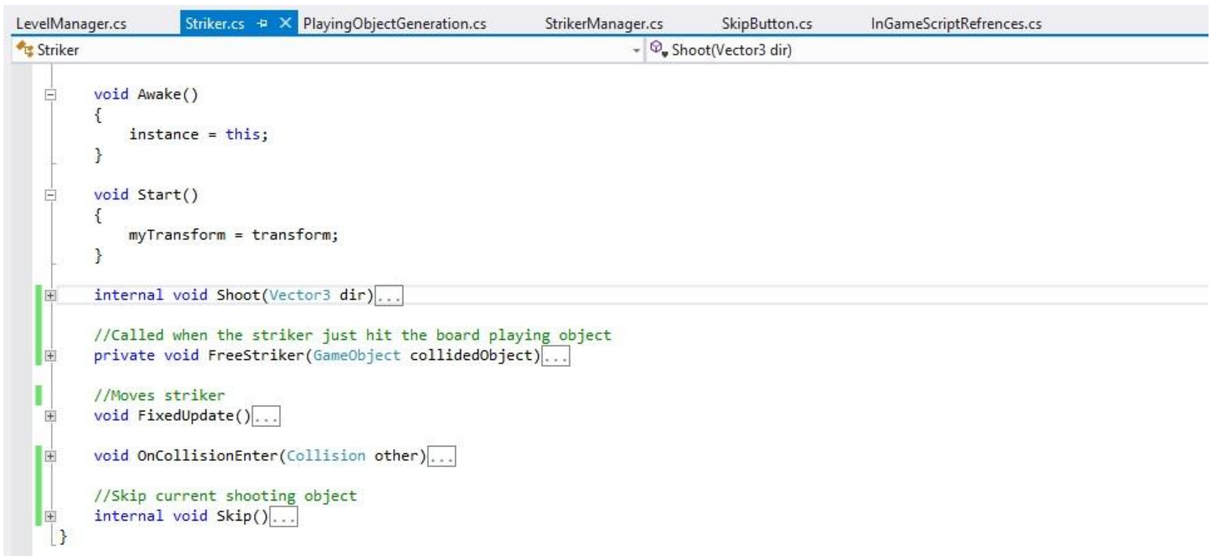
```
int numberOfRowsGenerated = 0;
float fallDownTime;
float currentRowAddingInterval;

void FalsenIsStarting()...

//Ensures that there is atleast a minimum no of rows in the screen
internal void CheckForMinRowCount(GameObject bottomMostObject)...

void InitiateRowAdd()...

//Adds new row at top of the screen and move the objects down by rowgap
IEnumerator AddRow()...

internal int GetObjectCount()...


//If any of the objects touches threshold line. The game is over.
private void CheckForGameOver()...
```

PlayingObjectManager.cs

This script is responsible for all the major operation realted to Playing Objects.

Striker.cs

This script holds the current shooting object and its related operations.



```
    void Awake()
    {
        instance = this;
    }

    void Start()
    {
        myTransform = transform;
    }

    internal void Shoot(Vector3 dir)...

    //Called when the striker just hit the board playing object
    private void FreeStriker(GameObject collidedObject)...

    //Moves striker
    void FixedUpdate()...

    void OnCollisionEnter(Collision other)...

    //Skip current shooting object
    internal void Skip()...
}
```