Implementation

4.1 Game Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameController : MonoBehaviour
{
public int row, col, countStep;
public int level;
public int rowBlank, colBlank;
//position of Image Blank
public int sizeRow, sizeCol;

int countPoint = 0;
int countImageKey = 0;
int countComplete = 0;

public bool startControl = false;
public bool checkComplete;
public bool gameIsComplete;
public string winScreen;


GameObject temp;
public List<GameObject> imageKeyList; //run
form 0 --> list.count
public List<GameObject>
imageOfPictureList;
public List<GameObject> checkPointList;

GameObject[,] imageKeyMatrix;
GameObject[,] imageOfPictureMatrix;
GameObject[,] checkPointMatrix;
// Use this for initialization

void Start () {
imageKeyMatrix = new
GameObject[sizeRow, sizeCol];
imageOfPictureMatrix = new
GameObject[sizeRow, sizeCol];
checkPointMatrix = new
GameObject[sizeRow, sizeCol];
if (level == 1)
{
imageOfEasyLevel();
}
else if (level == 2)
{
imageOfNormalLevel();
}
else if (level == 3){
imageOfHardLevel();}
checkPointManager();
ImageKeyManager();
for (int r = 0; r < sizeRow; r++)
{ //run row
for (int c = 0; c < sizeCol;c++)
{//run col
if(imageOfPictureMatrix[r,c].name.
CompareTo("blank") == 0){
rowBlank = r;
colBlank = c;
break;
}}}}

void checkPointManager(){
for (int r = 0; r < sizeRow; r++){ //run row
for (int c = 0; c < sizeCol; c++){//run col
checkPointMatrix[r, c] =
checkPointList[countPoint];
countPoint++;
}}}

void ImageKeyManager(){
for (int r = 0; r < sizeRow; r++){ //run row
for (int c = 0; c < sizeCol; c++){//run col
imageKeyMatrix[r, c] =
imageKeyList[countImageKey];
countImageKey++;}}}

// Update is called once per frame
void Update () {
if (startControl)
{ // move for image of puzzle
startControl = false;
if (countStep==1) {
if (imageOfPictureMatrix[row, col] != null
&& imageOfPictureMatrix[row,col].name.
CompareTo("blank") != 0)
{ // Check if image touch is blank
if(rowBlank!=row && colBlank == col)
{
if (Mathf.Abs(row - rowBlank) == 1)
{
//move
//call Function Imagesort
sortImage();
countStep = 0; //reset Count Step
}
else
{
countStep = 0;
}
}
else if(rowBlank==row && colBlank != col)
{
if (Mathf.Abs(col - colBlank) == 1)
{
//move

sortImage();
countStep = 0;
}
else
{
countStep = 0;
}
}
else if ((rowBlank == row && colBlank ==
col) || (rowBlank!=row && colBlank!=col))
{
countStep = 0; // not move
}
}
else
{
countStep = 0; //not move
}
}
}
}

private void FixedUpdate()
{
if (checkComplete)
{
checkComplete = false;
for(int r = 0; r < sizeRow; r++)
{
for(int c = 0; c < sizeCol; c++)
{
if (imageKeyMatrix[r,
c].gameObject.name.CompareTo(imageOfPic
tureMatrix[r, c].gameObject.name) == 0)
{
countComplete++;}
else{
break;
}}}
if (countComplete == checkPointList.Count)
{gameIsComplete = true;
Debug.Log("Game is complete");
SceneManager.LoadScene(winScreen);
}
else{
countComplete = 0;
}}}

void sortImage(){
temp = imageOfPictureMatrix[rowBlank,
colBlank];
imageOfPictureMatrix[rowBlank, colBlank] =
null;
imageOfPictureMatrix[rowBlank, colBlank] =
imageOfPictureMatrix[row, col]; //select
image is not image blank and save it at
position
imageOfPictureMatrix[row, col] = null;
imageOfPictureMatrix[row, col] = temp;

//move image
imageOfPictureMatrix[rowBlank,
colBlank].GetComponent<ImageController>(
).
target = checkPointMatrix[rowBlank,
colBlank];//set new point for image blank
imageOfPictureMatrix[row,col].
GetComponent<ImageController>().target =
checkPointMatrix[row, col];

imageOfPictureMatrix[rowBlank,colBlank].
GetComponent<ImageController>().startMo
ve = true;
imageOfPictureMatrix[row,col].GetCompon
ent<ImageController>().startMove = true;

//set row and col for blank image
```

```
rowBlank = row;
colBlank = col;}

void imageOfEasyLevel()
{
imageOfPictureMatrix[0, 0] =
imageOfPictureList[0];
imageOfPictureMatrix[0, 1] =
imageOfPictureList[2];
imageOfPictureMatrix[0, 2] =
imageOfPictureList[5];
imageOfPictureMatrix[1, 0] =
imageOfPictureList[4];
imageOfPictureMatrix[1, 1] =
imageOfPictureList[1];
imageOfPictureMatrix[1, 2] =
imageOfPictureList[7];
imageOfPictureMatrix[2, 0] =
imageOfPictureList[3];
imageOfPictureMatrix[2, 1] =
imageOfPictureList[6];
imageOfPictureMatrix[2, 2] =
imageOfPictureList[8];
}

void imageOfNormalLevel()
{
imageOfPictureMatrix[0, 0] =
imageOfPictureList[4];
imageOfPictureMatrix[0, 1] =
imageOfPictureList[0];
imageOfPictureMatrix[0, 2] =
imageOfPictureList[1];
imageOfPictureMatrix[0, 3] =
imageOfPictureList[2];
imageOfPictureMatrix[1, 0] =
imageOfPictureList[8];
imageOfPictureMatrix[1, 1] =
imageOfPictureList[6];
imageOfPictureMatrix[1, 2] =
imageOfPictureList[7];
imageOfPictureMatrix[1, 3] =
imageOfPictureList[11];
imageOfPictureMatrix[2, 0] =
imageOfPictureList[12];
imageOfPictureMatrix[2, 1] =
imageOfPictureList[5];
imageOfPictureMatrix[2, 2] =
imageOfPictureList[14];
imageOfPictureMatrix[2, 3] =
imageOfPictureList[10];
imageOfPictureMatrix[3, 0] =
imageOfPictureList[13];
imageOfPictureMatrix[3, 1] =
imageOfPictureList[9];
imageOfPictureMatrix[3, 2] =
imageOfPictureList[15];
imageOfPictureMatrix[3, 3] =
imageOfPictureList[3];
}

void imageOfHardLevel()
{
imageOfPictureMatrix[0, 0] =
imageOfPictureList[5];
imageOfPictureMatrix[0, 1] =
imageOfPictureList[2];
imageOfPictureMatrix[0, 2] =
imageOfPictureList[3];
imageOfPictureMatrix[0, 3] =
imageOfPictureList[4];
imageOfPictureMatrix[0, 4] =
imageOfPictureList[9];
imageOfPictureMatrix[1, 0] =
imageOfPictureList[10];
imageOfPictureMatrix[1, 1] =
imageOfPictureList[1];
imageOfPictureMatrix[1, 2] =
imageOfPictureList[12];
imageOfPictureMatrix[1, 3] =
imageOfPictureList[7];
imageOfPictureMatrix[1, 4] =
imageOfPictureList[8];
imageOfPictureMatrix[2, 0] =
imageOfPictureList[15];
imageOfPictureMatrix[2, 1] =
imageOfPictureList[6];
imageOfPictureMatrix[2, 2] =
imageOfPictureList[13];
imageOfPictureMatrix[2, 3] =
imageOfPictureList[14];
imageOfPictureMatrix[2, 4] =
imageOfPictureList[19];
imageOfPictureMatrix[3, 0] =
imageOfPictureList[20];
imageOfPictureMatrix[3, 1] =
imageOfPictureList[11];
imageOfPictureMatrix[3, 2] =
imageOfPictureList[22];
imageOfPictureMatrix[3, 3] =
imageOfPictureList[17];
```

```
imageOfPictureMatrix[3, 4] =
imageOfPictureList[18];
imageOfPictureMatrix[4, 0] =
imageOfPictureList[21];
imageOfPictureMatrix[4, 1] =
imageOfPictureList[16];
imageOfPictureMatrix[4, 2] =
imageOfPictureList[23];
imageOfPictureMatrix[4, 3] =
imageOfPictureList[24];
imageOfPictureMatrix[4, 4] =
imageOfPictureList[0];
}}
```

4.2 Level Manager

```
using UnityEngine;
using System.Collections;
using UnityEngine.SceneManagement;

public class LevelManager :
MonoBehaviour {

public void LoadLevel(string name){
Debug.Log("New Level load:"+ name);
SceneManager.LoadScene (name);}

public void QuitRequest(){
Debug.Log ("Quit requested");
Application.Quit();}
}
```

4.3 Step By Step Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StepByStepController :
MonoBehaviour {
public int row, col;
GameController gameMN;
    // Use this for initialization
void Start () {
GameObject gamemanager =
GameObject.Find("GameController");
gameMN =
gamemanager.GetComponent<GameController>
();}

       // Update is called once per frame
void Update () {}

private void OnMouseDown()
{Debug.Log("Row is :" + row + "Col is :"
+ col);
        gameMN.countStep += 1;
        gameMN.row = row;
        gameMN.col = col;
        gameMN.startControl = true;
    }
}
```