**プログラミング概論２　自主開発課題　レポート**

19238901　明石 華実

* **開発したものの名前**

BROCK BREAKER

* **目的／用途**

サンプルより，実用性，安全性を考慮し，作成しました．

* **開発環境と使用言語**

C#

* **動作環境**

Windows

* **アプリケーションの外観**

|  |  |
| --- | --- |
| グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明 |  |
| **図１　ログイン画面** | **図2　新規ユーザー登録** |
|  |  |
|  |  |
| **図３　ホーム画面** | **図4　ランキング画面** |
|  |  |
|  |  |
| **図5　戦績画面** | **図6　画面/ゲーム説明画面** |
|  |  |
|  | テーブル  低い精度で自動的に生成された説明 |
| **図7****ゲーム画面** | **図８　リザルト画面(CLEAR/GAME OVER)** |

* **使い方の説明**

初めに，このアプリでは，会員登録/ゲーム記録の際に，csvファイルを使用しています．

＜ゲーム説明＞

* ゲーム画面の色付きのプレイ画面上で反射する赤いボールを操って，プレイ画面に表示される紫色のブロックを崩すゲーム．
* プレイ画面の底辺に赤いボールが落下すると，ゲームオーバーとなり，リザルト画面に移る．
* プレイ画面に表示された紫色のブロックをすべて崩すことができると，ゲームクリアとなり，リザルト画面に移る．
* ボタンで選択したレベルのゲームをプレイすることができる．

＜画面/ボタン説明＞

1. ログイン画面

[画面説明]

* ユーザー登録している人は，IDとパスワードを入力することで，ログインをすることができる．

[ボタン説明]

* [LOG IN]ボタンを押すことで，IDとパスワードが一致する場合，ログインし，ホーム画面に遷移する．

ID/パスワードが未入力，もしくは，ID/パスワードが登録情報と異なる場合，警告を表示する．

* [NEW PLAYER]ボタンを押すことで，新規ユーザー登録画面に遷移する．
* [GUEST PLAY]ボタンを押すことで，ゲストとしてログインすることができる．

ゲストとしてログインすると，一部の機能が使えなくなる．

1. 新規ユーザー登録

[画面説明]

* 新規ユーザー登録を行うことができる．

[ボタン説明]

* [LOG IN]ボタンを押すことで，ログイン画面に遷移する．
* [REGISTRATION]ボタンを押すことで，ID/氏名/パスワード/地域が正しく入力されている場合，新規ユーザーとして，情報を登録する．

ID/氏名/パスワード/地域が未入力，もしくは，ID/氏名/パスワード/地域が誤って入力されている場合，警告を表示する．

* [ALL CLEAR]ボタンを押すことで，ユーザー情報をすべて削除する．

1. ホーム画面

[画面説明]

* ログイン後，様々な機能を利用することができる．

[ボタン説明]

* [GAME]ボタンを押すことで，ゲームプレイ画面に遷移する．
* [RANKING]ボタンを押すことで，ランキング画面に遷移する．
* [STATES]ボタンを押すことで，ユーザー戦績画面に遷移する．

※ゲストプレイでは，戦績が残らないため利用できない．

* [MANUAL]ボタンを押すことで，画面/ゲーム説明画面に遷移する．
* [LOG OUT]ボタンを押すことで，ログアウトし，ログイン画面に遷移する．

1. ランキング画面

[画面説明]

* ゲームの各レベルのランキングを表示することができる．
* ランキングは，以下のルールに従って表示される．
* ゲームクリアしたプレイヤーは“黒文字”，ゲームオーバーだったプレイヤーは“青文字”，プレイされた記録がない場合は“赤文字”で表記する．
* ランキング上位の優先度は，以下のルールに従う．
  + 1. ゲームクリアしたプレイヤーの高得点順
    2. 1.がいない場合，ゲームオーバーしたプレイヤーの高得点順
    3. 2.がいない場合，“-”を表示

[ボタン説明]

* [HOME]ボタンを押すことで，ホーム画面に遷移する．
* [GAME]ボタンを押すことで，ゲームプレイ画面に遷移する．
* [LEVEL1]~[LEVEL7]ボタンを押すことで，各レベルのランキング画面を表示することができる．

1. 戦績画面

[画面説明]

* ゲームの各レベルのユーザー戦績（上位3つ）を表示することができる．
* ユーザー戦績は，以下のルールに従って表示される．
* ゲームクリアした回の戦績は“黒文字”，ゲームオーバーだった回の戦績は“青文字”，プレイされた記録がない場合は“赤文字”で表記する．
* ランキング上位の優先度は，以下のルールに従う．
  + 1. ゲームクリアした回の高得点順
    2. 1.がいない場合，ゲームオーバーした回の高得点順
    3. 2.がいない場合，“-”を表示

[ボタン説明]

* [HOME]ボタンを押すことで，ホーム画面に遷移する．
* [GAME]ボタンを押すことで，ゲームプレイ画面に遷移する．
* [LOG OUT]ボタンを押すことで，ログアウトし，ログイン画面に遷移する．
* [LEVEL1]~[LEVEL7]ボタンを押すことで，各レベルのユーザー戦績画面を表示することができる．

1. 画面/ゲーム説明画面

[画面説明]

* ゲームのユーザーインターフェース（UI）と，各機能/画面の説明を表示することができる．

[ボタン説明]

* [HOME]ボタンを押すことで，ホーム画面に遷移する．
* [GAME]ボタンを押すことで，ゲームプレイ画面に遷移する．
* [LOG IN]ボタンを押すことで，ログイン画面のUIと説明を表示する．
* [REGISTRATION]ボタンを押すことで，新規ユーザー登録画面のUIと説明を表示する．
* [HOME]ボタンを押すことで，ホーム画面のUIと説明を表示する．
* [RANKING]ボタンを押すことで，ランキング画面のUIと説明を表示する．
* [RESULT]ボタンを押すことで，リザルト画面のUIと説明を表示する．
* [STATS]ボタンを押すことで，ユーザー戦績画面のUIと説明を表示する．

1. ゲーム画面

[画面説明]

* 選択したレベルのゲームをプレイすることができる．

[ボタン説明]

* [START]ボタンを押すことで，ゲームが開始される．
* [STOP]ボタンを押すことで，ゲームが一時停止される．
* [RESTART]ボタンを押すことで，ゲームが再開される．
* [RESET]ボタンを押すことで，ゲームがリセットされる．
* [RANKING]ボタンを押すことで，ランキング画面に遷移する．
* [AUTO]ボタンを押すことで，オートでゲームプレイ画面下部のバーを自動で操作する．

※オートモードにすると，スコアが減る．

※クリアするためのモードではない．

* [HOME]ボタンを押すことで，ホーム画面に遷移する．
* [LEVEL1]~[LEVEL7]ボタンを押すことで，ゲームのレベルを選択する．

何も入力されなかった場合，LEVEL1となる．

1. リザルト画面

[画面説明]

* ゲームの結果を表示する．

[ボタン説明]

* [FORWARD]ボタンを押すことで，自分が選択し，プレイしたゲームの1つ下のレベルのゲームが開始される．

※LEVEL1のゲームをプレイしていた場合，LEVEL7のゲームが始まる．

* [REGAME]ボタンを押すことで，自分が選択し，プレイしたゲームと同じレベルのゲームが開始される．
* [NEXT]ボタンを押すことで，自分が選択し，プレイしたゲームの1つ上のレベルのゲームが開始される．
* ※LEVEL7のゲームをプレイしていた場合，LEVEL１のゲームが始まる．
* [RANKING]ボタンを押すことで，ランキング画面に遷移する．
* [HOME]ボタンを押すことで，ホーム画面に遷移する．
* **実装内容／改良部分**

1. ログイン画面

**<Page4.xaml>**

1. <Page x:Class="Game2.Page4"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d"
8. d:DesignHeight="450" d:DesignWidth="800"
9. Title="Page4" HorizontalAlignment="Center">
10. <Grid Background="White" HorizontalAlignment="Center" VerticalAlignment="Center" Height="450" Width="800">
11. <Label Content="BRICK BREAKER" Margin="0,6,0,0" FontSize="72" HorizontalAlignment="Center" FontWeight="Bold" FontFamily="Broadway" Width="665" Height="93" VerticalAlignment="Top"/>
12. <Grid Margin="0,100,0,0" HorizontalAlignment="Center" VerticalAlignment="Top" Width="600">
13. <Label Content="ID:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="200" FontSize="60" FontFamily="Broadway"/>
14. <TextBox x:Name="Player\_Id" HorizontalAlignment="Right" Height="78" Margin="207,0,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="390" SelectionBrush="#FF71B9F2" FontSize="48"/>
15. </Grid>
16. <Grid Margin="0,180,0,0" HorizontalAlignment="Center" VerticalAlignment="Top" Width="600">
17. <Label Content="PASS:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="200" FontSize="60" FontFamily="Broadway"/>
18. <TextBox x:Name="Player\_Pass" Height="78" Margin="207,0,0,0" TextWrapping="Wrap" VerticalAlignment="Top" SelectionBrush="#FF71B9F2" FontSize="48" Width="390" HorizontalAlignment="Right"/>
19. </Grid>
20. <Grid HorizontalAlignment="Center" Height="150" Margin="0,290,0,0" VerticalAlignment="Top" Width="665">
21. <Button Content="LOG IN" Height="146" Margin="0,0,0,0" VerticalAlignment="Top" Width="200" Click="Button\_Click" HorizontalAlignment="Left"/>
22. <Button Content="NEW PLAYER" HorizontalAlignment="Center" Height="145" Margin="0,0,0,0" VerticalAlignment="Top" Width="200" Click="Button\_Click\_2"/>
23. <Button Content="GUEST PLAY" HorizontalAlignment="Right" Height="145" Margin="420,0,0,0" VerticalAlignment="Top" Width="200" Click="Button\_Click\_1"/>
24. </Grid>
25. </Grid>
26. </Page>

**<Page4.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.IO;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. using System.Windows;
8. using System.Windows.Controls;
9. using System.Windows.Data;
10. using System.Windows.Documents;
11. using System.Windows.Input;
12. using System.Windows.Media;
13. using System.Windows.Media.Imaging;
14. using System.Windows.Navigation;
15. using System.Windows.Shapes;
16. namespace Game2
17. {
18. /// <summary>
19. /// Page1.xaml の相互作用ロジック
20. /// </summary>
21. public partial class Page4 : Page
22. {
23. public class user
24. {
25. public string id { get; set; }
26. public string name { get; set; }
27. public string pass { get; set; }
28. public string from { get; set; }
29. public double score { get; set; }
30. }
31. List<user> U = new List<user>();
32. public Page4()
33. {
34. InitializeComponent();
35. try
36. {
37. var filePath = "..\\test.csv";
38. // 読み込みたいCSVファイルのパスを指定して開く
39. StreamReader sr = new StreamReader(filePath);
40. {
41. // 末尾まで繰り返す
42. while (!sr.EndOfStream)
43. {
44. // CSVファイルの一行を読み込む
45. string line = sr.ReadLine();
46. // 読み込んだ一行をカンマ毎に分けて配列に格納する
47. string[] values = line.Split(',');
48. user nu = new user();
49. nu.id = values[0];
50. nu.name = values[1];
51. nu.pass = values[2];
52. nu.from = values[3];
53. nu.score = Double.Parse(values[4]);
54. U.Add(nu);
55. }
56. }
57. }
58. catch (Exception e)
59. {
60. Console.WriteLine(e.Message);
61. }
62. }
63. //LOG IN
64. private void Button\_Click(object sender, RoutedEventArgs e)
65. {
66. double ha = 0;
67. string care = "";
68. if (Player\_Id.Text == "")
69. {
70. care = care + "Enter your ID.\n";
71. Player\_Id.BorderBrush = Brushes.Red;
72. ha = 1;
73. }
74. else
75. {
76. Player\_Id.BorderBrush = Brushes.Black;
77. }
78. if (Player\_Pass.Text == "")
79. {
80. care = care + "Enter your Pass.\n";
81. Player\_Pass.BorderBrush = Brushes.Red;
82. ha = 2;
83. }
84. else
85. {
86. Player\_Pass.BorderBrush = Brushes.Black;
87. }
88. if (ha != 0)
89. {
90. MessageBox.Show(care);
91. return;
92. }
93. List<user> u4n = U.FindAll(u => u.id == Player\_Id.Text);
94. if(u4n.Count == 0)
95. {
96. MessageBox.Show("Different ID.\n");
97. Player\_Id.Text = "";
98. Player\_Pass.Text = "";
99. Player\_Pass.BorderBrush = Brushes.Red;
100. Player\_Id.BorderBrush = Brushes.Red;
101. }
102. else
103. {
104. if (U.Find(u => u.id == Player\_Id.Text).pass == Player\_Pass.Text)
105. {
106. var page1 = new Page1(U.Find(u => u.id == Player\_Id.Text).name, U.Find(u => u.id == Player\_Id.Text).from, 1);
107. NavigationService.Navigate(page1);
108. }
109. else
110. {
111. MessageBox.Show("Different Pass.\n");
112. Player\_Pass.Text = "";
113. Player\_Pass.BorderBrush = Brushes.Red;
114. Player\_Id.BorderBrush = Brushes.Black;
115. }
116. }
117. }
118. //GUEST PLAY
119. private void Button\_Click\_1(object sender, RoutedEventArgs e)
120. {
121. var page1 = new Page1("Guest", "-", 1);
122. NavigationService.Navigate(page1);
123. }
124. //NEW PLAYER
125. private void Button\_Click\_2(object sender, RoutedEventArgs e)
126. {
127. var page5 = new Page5(U);
128. NavigationService.Navigate(page5);
129. }
130. }
131. }
132. 新規ユーザー登録

**<Page5.xaml>**

1. <Page x:Class="Game2.Page5"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d" d:DesignWidth="973.349"
8. Title="Page5" HorizontalAlignment="Center" VerticalAlignment="Center" Height="590">
9. <Grid Background="White" Margin="0,0,0,0">
10. <Label Content="USER REGISTRATION" Margin="0,3,0,0" FontSize="80" HorizontalAlignment="Center" FontWeight="Bold" FontFamily="Broadway" Width="924" Height="86" VerticalAlignment="Top"/>
11. <Grid HorizontalAlignment="Center" VerticalAlignment="Top" Width="708" Margin="0, 100, 0,0">
12. <Label Content="ID:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="220" FontSize="60" FontFamily="Broadway" Margin="0,0,0,0"/>
13. <TextBox x:Name="Player\_Id" HorizontalAlignment="Left" Height="78" Margin="250,0,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="450" SelectionBrush="#FF71B9F2" FontSize="48"/>
14. </Grid>
15. <Grid HorizontalAlignment="Center" VerticalAlignment="Top" Width="708" Margin="0, 175, 0,0">
16. <Label Content="NAME:" Margin="0,0,0,0" HorizontalAlignment="Left" Height="78" Width="220" FontSize="60" FontFamily="Broadway"/>
17. <TextBox x:Name="Player\_Name" HorizontalAlignment="Left" Height="78" Margin="250,0,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="450" SelectionBrush="#FF71B9F2" FontSize="48"/>
18. </Grid>
19. <Grid HorizontalAlignment="Center" Height="140" Margin="0,430,0,0" VerticalAlignment="Top" Width="620">
20. <Button Content="LOG IN" HorizontalAlignment="Left" Height="140" VerticalAlignment="Top" Width="200" Click="Button\_Click"/>
21. <Button Content="REGISTRATION" Height="140" VerticalAlignment="Top" Click="Button\_Click\_2" HorizontalAlignment="Center" Width="200"/>
22. <Button Content="ALL CLEAR" HorizontalAlignment="Right" Height="140" VerticalAlignment="Center" Width="200" Click="Button\_Click\_1"/>
23. </Grid>
24. <Grid HorizontalAlignment="Center" Margin="0,325,0,0" VerticalAlignment="Top" Width="708">
25. <Label Content="FROM:" HorizontalAlignment="Left" VerticalAlignment="Center" Height="78" Width="220" FontSize="60" FontFamily="Broadway"/>
26. <TextBox x:Name="Player\_From" Height="78" Margin="250,0,0,0" TextWrapping="Wrap" VerticalAlignment="Center" SelectionBrush="#FF71B9F2" FontSize="48" HorizontalAlignment="Left" Width="450"/>
27. </Grid>
28. <Grid HorizontalAlignment="Center" Margin="0,250,0,0" VerticalAlignment="Top" Width="708">
29. <Label Content="PASS:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="220" FontSize="60" FontFamily="Broadway"/>
30. <TextBox x:Name="Player\_Pass" HorizontalAlignment="Left" Height="78" Margin="250,0,0,0" TextWrapping="Wrap" VerticalAlignment="Center" Width="450" SelectionBrush="#FF71B9F2" FontSize="48"/>
31. </Grid>
32. </Grid>
33. </Page>

**<Page5.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.IO;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. using System.Windows;
8. using System.Windows.Controls;
9. using System.Windows.Data;
10. using System.Windows.Documents;
11. using System.Windows.Input;
12. using System.Windows.Media;
13. using System.Windows.Media.Imaging;
14. using System.Windows.Navigation;
15. using System.Windows.Shapes;
16. namespace Game2
17. {
18. /// <summary>
19. /// Page5.xaml の相互作用ロジック
20. /// </summary>
22. public partial class Page5 : Page
23. {
24. List<Page4.user> U = new List<Page4.user>();
25. public Page5(List<Page4.user> u4)
26. {
27. InitializeComponent();
28. this.U = u4;
29. }
30. //LOG IN
31. private void Button\_Click(object sender, RoutedEventArgs e)
32. {
33. var page4 = new Page4();
34. NavigationService.Navigate(page4);
35. }
36. //REGISTRATION
37. private void Button\_Click\_2(object sender, RoutedEventArgs e)
38. {
39. Player\_Id.BorderBrush = Brushes.Black;
40. Player\_Name.BorderBrush = Brushes.Black;
41. Player\_From.BorderBrush = Brushes.Black;
42. Player\_Pass.BorderBrush = Brushes.Black;
43. string care = "";
44. int h = 0;
45. if(Player\_Id.Text != "")
46. {
47. if (Player\_Id.Text.Contains(",") || Player\_Id.Text.Contains(" "))
48. {
49. care = care + "Can't use [,] or [ ] for ID.\n";
50. Player\_Id.Text = "";
51. Player\_Id.BorderBrush = Brushes.Red;
52. h = 1;
53. }
54. if (U.Any(u => u.id == Player\_Id.Text))
55. {
56. care = care + "The ID entered is already in use.\nEnter a different ID.\n";
57. Player\_Id.Text = "";
58. Player\_Id.BorderBrush = Brushes.Red;
59. h = 1;
60. }
61. if (Player\_Id.Text == "Guest")
62. {
63. care = care + "Can't use that ID.\nEnter a different ID.\n";
64. Player\_Id.Text = "";
65. Player\_Id.BorderBrush = Brushes.Red;
66. h = 1;
67. }
68. }
69. else
70. {
71. if (Player\_Id.Text == "")
72. {
73. Player\_Id.BorderBrush = Brushes.Red;
74. care = care + "Enter the ID\n";
75. h = 1;
76. }
77. }
78. if(Player\_Name.Text != "")
79. {
80. if (Player\_Name.Text.Contains(",") || Player\_Name.Text.Contains(" "))
81. {
82. care = care + "Can't use [,] or [ ] for NAME.\n";
83. Player\_Name.BorderBrush = Brushes.Red;
84. Player\_Name.Text = "";
85. h = 1;
86. }
87. if (U.Any(u => u.name == Player\_Name.Text))
88. {
89. care = care + "The NAME entered is already in use.\nEnter a different NAME.\n";
90. Player\_Name.Text = "";
91. Player\_Name.BorderBrush = Brushes.Red;
92. h = 1;
93. }
94. if (Player\_Name.Text == "Guest")
95. {
96. care = care + "Can't use that NAME.\nEnter a different NAME.\n";
97. Player\_Name.Text = "";
98. Player\_Name.BorderBrush = Brushes.Red;
99. h = 1;
100. }
101. }
102. else
103. {
104. if (Player\_Name.Text == "")
105. {
106. Player\_Name.BorderBrush = Brushes.Red;
107. care = care + "Enter the NAME\n";
108. h = 1;
109. }
110. }
111. if(Player\_Pass.Text != "")
112. {
113. if (Player\_From.Text.Contains(",") || Player\_From.Text.Contains(" "))
114. {
115. care = care + "Can't use [,] or [ ] for FROM.\n";
116. Player\_From.Text = "";
117. Player\_From.BorderBrush = Brushes.Red;
118. h = 1;
119. }
120. }
121. else
122. {
123. if (Player\_Pass.Text == "")
124. {
125. Player\_Pass.BorderBrush = Brushes.Red;
126. care = care + "Enter the PASS\n";
127. h = 1;
128. }
129. }
130. if(Player\_From.Text != "")
131. {
132. if (Player\_Pass.Text.Contains(",") || Player\_Pass.Text.Contains(" "))
133. {
134. care = care + "Can't use [,] or [ ] for PASS.\n";
135. Player\_Pass.BorderBrush = Brushes.Red;
136. Player\_Pass.Text = "";
137. h = 1;
138. }
139. }
140. else
141. {
142. if (Player\_From.Text == "")
143. {
144. Player\_From.BorderBrush = Brushes.Red;
145. care = care + "Enter the FROM\n";
146. h = 1;
147. }
148. }
149. if (h != 0)
150. {
151. MessageBox.Show(care);
152. return;
153. }
154. if (Player\_Id.Text != "" && Player\_Name.Text != "" && Player\_Pass.Text != "" && Player\_From.Text != "")
155. {
156. var filePath = "..\\test.csv";
158. string[] values = { Player\_Id.Text, Player\_Name.Text, Player\_Pass.Text, Player\_From.Text, 0.ToString() };
159. using (var sw = new System.IO.StreamWriter(filePath, true))
160. {
161. sw.WriteLine("{0},{1},{2},{3},{4}", values[0], values[1], values[2], values[3], values[4]);
162. }
163. MessageBox.Show("Registration complete!\nPlayer INFO:\nID:" + Player\_Id.Text + "\nName:" + Player\_Name.Text + "\nPass:" + Player\_Pass.Text + "\nFrom:" + Player\_From.Text);
164. Player\_Id.Text = "";
165. Player\_Name.Text = "";
166. Player\_Pass.Text = "";
167. Player\_From.Text = "";
168. return;
169. }
170. MessageBox.Show(care);
171. }
172. //ALL CLEAR
173. private void Button\_Click\_1(object sender, RoutedEventArgs e)
174. {
175. Player\_Id.Text = "";
176. Player\_Name.Text = "";
177. Player\_Pass.Text = "";
178. Player\_From.Text = "";
179. }
180. }
181. }
182. ホーム画面

**<Page1.xaml>**

1. <Page x:Class="Game2.Page1"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d"
8. Title="Page1" Height="459" Width="741">
9. <Grid Background="White" HorizontalAlignment="Center" VerticalAlignment="Center" Height="459" Width="741">
10. <Grid.ColumnDefinitions>
11. <ColumnDefinition Width="5\*"/>
12. <ColumnDefinition Width="1042\*"/>
13. </Grid.ColumnDefinitions>
14. <Label Content="BRICK BREAKER" FontSize="72" HorizontalAlignment="Center" FontWeight="Bold" FontFamily="Broadway" Grid.Column="1" Width="666" Margin="0,0,0,366"/>
15. <Grid Margin="0,180,0,0" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Top" Width="576">
16. <Label Content="FROM:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
17. <TextBox x:Name="Player\_from" HorizontalAlignment="Right" Height="78" Margin="249,0,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
18. </Grid>
19. <Grid Grid.Column="1" HorizontalAlignment="Center" Margin="0,101,0,0" VerticalAlignment="Top" Width="576">
20. <Label Content="NAME:" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
21. <TextBox x:Name="Player\_Name" HorizontalAlignment="Right" Height="78" Margin="249,0,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
22. </Grid>
23. <Grid Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,300,0,0" Width="690">
24. <Button Content="GAME" HorizontalAlignment="Left" Height="145" VerticalAlignment="Top" Width="130" Click="Button\_Click"/>
25. <Button Content="RANKING" HorizontalAlignment="Left" Height="145" Margin="140,0,0,0" VerticalAlignment="Top" Width="130" Click="Button\_Click\_2"/>
26. <Button Content="MANUAL" HorizontalAlignment="Left" Height="145" Margin="420,0,0,0" VerticalAlignment="Top" Width="130" Click="Button\_Click\_3"/>
27. <Button Content="STATES" HorizontalAlignment="Left" Height="145" Margin="280,0,0,0" VerticalAlignment="Top" Width="130" Click="Button\_Click\_States"/>
28. <Button Content="LOG OUT" HorizontalAlignment="Left" Height="145" Margin="560,0,0,0" VerticalAlignment="Top" Width="130" Click="Button\_Click\_1"/>
29. </Grid>
30. </Grid>
31. </Page>

**<Page1.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. using System.Windows;
7. using System.Windows.Controls;
8. using System.Windows.Data;
9. using System.Windows.Documents;
10. using System.Windows.Input;
11. using System.Windows.Media;
12. using System.Windows.Media.Imaging;
13. using System.Windows.Navigation;
14. using System.Windows.Shapes;
15. namespace Game2
16. {
17. /// <summary>
18. /// Page1.xaml の相互作用ロジック
19. /// </summary>
20. public partial class Page1 : Page
21. {
22. double p\_level1 = 1;
23. public Page1(string p\_name, string p\_from, double p\_level)
24. {
25. InitializeComponent();
26. Player\_Name.Text = p\_name;
27. Player\_Name.IsReadOnly = true;
28. Player\_from.Text = p\_from;
29. Player\_from.IsReadOnly = true;
30. p\_level1 = p\_level;
31. }
32. private void Button\_Click(object sender, RoutedEventArgs e)
33. {
34. var page2 = new Page2(Player\_Name.Text, Player\_from.Text, p\_level1);
35. NavigationService.Navigate(page2);
36. }
37. //LOG OUT
38. private void Button\_Click\_1(object sender, RoutedEventArgs e)
39. {
40. var page4 = new Page4();
41. NavigationService.Navigate(page4);
42. }
43. private void Button\_Click\_2(object sender, RoutedEventArgs e)
44. {
45. var page6 = new Page6(Player\_Name.Text, Player\_from.Text, p\_level1);
46. NavigationService.Navigate(page6);
47. }
48. //STATES
49. private void Button\_Click\_States(object sender, RoutedEventArgs e)
50. {
51. if(Player\_Name.Text == "Guest")
52. {
53. MessageBox.Show("Not available for [Guest Play].\nMembership registration is required to record/view your stats.\nBack [HOME].");
54. return;
55. }
56. var page7 = new Page7(Player\_Name.Text, Player\_from.Text, p\_level1);
57. NavigationService.Navigate(page7);
58. }
59. //MANUAL
60. private void Button\_Click\_3(object sender, RoutedEventArgs e)
61. {
62. var page8 = new Page8(Player\_Name.Text, Player\_from.Text, p\_level1);
63. NavigationService.Navigate(page8);
64. }
65. }
66. }
67. ランキング画面

**<Page6.xaml>**

1. <Page x:Class="Game2.Page6"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d" d:DesignWidth="1218.201"
8. Title="Page6" Height="840" HorizontalAlignment="Center" VerticalAlignment="Center">
9. <Grid Background="White" Margin="0,0,10,0" Height="840" VerticalAlignment="Top">
10. <Grid.RowDefinitions>
11. <RowDefinition Height="840\*"/>
12. <RowDefinition Height="\*"/>
13. </Grid.RowDefinitions>
14. <Label Content="BRICK BREAKER" Margin="0,10,0,0" FontSize="72" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Center" Width="668" VerticalAlignment="Top"/>
15. <Grid HorizontalAlignment="Center" Height="313" Margin="56,262,0,0" VerticalAlignment="Top" Width="1110">
16. <Label Content="No1:" HorizontalAlignment="Left" Margin="0,75,0,0" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
17. <Label Content="No2:" HorizontalAlignment="Left" Margin="0,155,0,0" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
18. <Label Content="No3:" HorizontalAlignment="Left" Margin="0,235,0,0" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
19. <TextBox x:Name="Player\_Name1" HorizontalAlignment="Left" Height="78" Margin="202,75,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
20. <TextBox x:Name="Player\_from1" HorizontalAlignment="Left" Height="78" Margin="532,75,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="246" SelectionBrush="#FF71B9F2" FontSize="48"/>
21. <TextBox x:Name="Player\_Score1" HorizontalAlignment="Left" Height="78" Margin="783,75,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
22. <TextBox x:Name="Player\_Name2" HorizontalAlignment="Left" Height="78" Margin="202,155,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
23. <TextBox x:Name="Player\_from2" HorizontalAlignment="Left" Height="78" Margin="532,155,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="246" SelectionBrush="#FF71B9F2" FontSize="48"/>
24. <TextBox x:Name="Player\_Score2" HorizontalAlignment="Left" Height="78" Margin="783,155,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
25. <TextBox x:Name="Player\_Name3" HorizontalAlignment="Left" Height="78" Margin="202,235,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
26. <TextBox x:Name="Player\_from3" HorizontalAlignment="Left" Height="78" Margin="532,235,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="246" SelectionBrush="#FF71B9F2" FontSize="48"/>
27. <TextBox x:Name="Player\_Score3" HorizontalAlignment="Left" Height="78" Margin="783,235,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
28. <Label Content="Name" Margin="258,0,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Left" Width="214" Height="70" VerticalAlignment="Top"/>
29. <Label Content="From" Margin="559,0,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Left" Width="192" Height="70" VerticalAlignment="Top"/>
30. <Label Content="Score" Margin="850,0,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Left" Width="192" Height="70" VerticalAlignment="Top"/>
31. <Label Content="Rank" HorizontalAlignment="Left" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
32. </Grid>
33. <Button Content="GAME" HorizontalAlignment="Center" Height="70" Margin="0,750,0,0" VerticalAlignment="Top" Width="770" Click="Button\_Click1"/>
34. <Button Content="HOME" Height="70" Margin="0,675,0,0" VerticalAlignment="Top" Click="Button\_Click" HorizontalAlignment="Center" Width="770"/>
35. <Label Content="RANKING" Margin="0,100,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Center" Width="324" Height="67" VerticalAlignment="Top"/>
36. <Grid Height="70" Margin="0,184,0,0" VerticalAlignment="Top" HorizontalAlignment="Center">
37. <TextBox x:Name="Player\_level" TextWrapping="Wrap" FontFamily="Broadway" FontSize="48" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="206,0,0,0" Height="62" VerticalAlignment="Center" HorizontalAlignment="Stretch"/>
38. <Label Content="Level:" FontSize="48" Background="{x:Null}" FontFamily="Broadway" Margin="0,0,222,0" Height="70" VerticalAlignment="Center"/>
39. </Grid>
40. <Grid Margin="0,590,0,0" Height="100" HorizontalAlignment="Center" VerticalAlignment="Top">
41. <Button Content="LEVEL1" Width="130" Click="Button\_Click\_lv1" Margin="0,0,0,0" HorizontalAlignment="Left" Height="80" VerticalAlignment="Top"/>
42. <Button Content="LEVEL2" Width="130" Click="Button\_Click\_lv2" Margin="140,0,0,0" HorizontalAlignment="Left" Height="80" VerticalAlignment="Top"/>
43. <Button Content="LEVEL3" Width="130" Click="Button\_Click\_lv3" Margin="280,0,0,0" HorizontalAlignment="Left" Height="80" VerticalAlignment="Top"/>
44. <Button Content="LEVEL4" Width="130" Click="Button\_Click\_lv4" Margin="420,0,0,0" HorizontalAlignment="Left" Height="80" VerticalAlignment="Top"/>
45. <Button Content="LEVEL5" Width="130" Click="Button\_Click\_lv5" Margin="560,0,0,0" HorizontalAlignment="Left" VerticalAlignment="Top" Height="80"/>
46. <Button Content="LEVEL6" Width="130" Click="Button\_Click\_lv6" Margin="700,0,0,0" HorizontalAlignment="Left" VerticalAlignment="Top" Height="80"/>
47. <Button Content="LEVEL7" Width="130" Click="Button\_Click\_lv7" Margin="840,0,0,0" HorizontalAlignment="Left" Height="80" VerticalAlignment="Top"/>
48. </Grid>
49. </Grid>
50. </Page>

**<Page6.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.IO;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. using System.Windows;
8. using System.Windows.Controls;
9. using System.Windows.Data;
10. using System.Windows.Documents;
11. using System.Windows.Input;
12. using System.Windows.Media;
13. using System.Windows.Media.Imaging;
14. using System.Windows.Navigation;
15. using System.Windows.Shapes;
16. namespace Game2
17. {
18. /// <summary>
19. /// Page6.xaml の相互作用ロジック
20. /// </summary>
21. public partial class Page6 : Page
22. {
23. public class user\_rank
24. {
25. public string name { get; set; }
26. public string from { get; set; }
27. public double level { get; set; }
28. public double score { get; set; }
29. public string result { get; set; }
30. }
31. string p\_name6 = "";
32. string p\_from6 = "";
33. double g\_level6 = 0;
34. public Page6(string p\_name, string p\_from, double g\_level)
35. {
36. InitializeComponent();
37. p\_from6 = p\_from;
38. p\_name6 = p\_name;
39. g\_level6 = g\_level;
40. Player\_level.IsReadOnly = true;
41. Player\_Name1.IsReadOnly = true;
42. Player\_Name2.IsReadOnly = true;
43. Player\_Name3.IsReadOnly = true;
44. Player\_from1.IsReadOnly = true;
45. Player\_from2.IsReadOnly = true;
46. Player\_from3.IsReadOnly = true;
47. Player\_Score1.IsReadOnly = true;
48. Player\_Score2.IsReadOnly = true;
49. Player\_Score3.IsReadOnly = true;
50. getRank(g\_level6);
51. }
52. public void getRank(double level)
53. {
54. Player\_level.Text = level.ToString();
55. List<user\_rank> u6 = new List<user\_rank>();
56. try
57. {
58. var filePath = "..\\result.csv";
59. // 読み込みたいCSVファイルのパスを指定して開く
60. StreamReader sr = new StreamReader(filePath);
61. {
62. // 末尾まで繰り返す
63. while (!sr.EndOfStream)
64. {
65. // CSVファイルの一行を読み込む
66. string line = sr.ReadLine();
67. // 読み込んだ一行をカンマ毎に分けて配列に格納する
68. string[] values = line.Split(',');
69. user\_rank nu = new user\_rank();
70. nu.name = values[0];
71. nu.from = values[1];
72. nu.level = Double.Parse(values[2]);
73. nu.score = Double.Parse(values[3]);
74. nu.result = values[4];
75. u6.Add(nu);
76. }
77. }
78. }
79. catch (Exception e)
80. {
81. Console.WriteLine(e.Message);
82. }
83. List<user\_rank> u\_level = u6.FindAll(u => u.level == level);
84. if (u\_level.Count == 0)
85. {
86. Player\_Name1.Text = "-";
87. Player\_Name1.Foreground = Brushes.Red;
88. Player\_from1.Text = "-";
89. Player\_from1.Foreground = Brushes.Red;
90. Player\_Score1.Text = "-";
91. Player\_Score1.Foreground = Brushes.Red;
92. Player\_Name2.Text = "-";
93. Player\_Name2.Foreground = Brushes.Red;
94. Player\_from2.Text = "-";
95. Player\_from2.Foreground = Brushes.Red;
96. Player\_Score2.Text = "-";
97. Player\_Score2.Foreground = Brushes.Red;
98. Player\_Name3.Text = "-";
99. Player\_Name3.Foreground = Brushes.Red;
100. Player\_from3.Text = "-";
101. Player\_from3.Foreground = Brushes.Red;
102. Player\_Score3.Text = "-";
103. Player\_Score3.Foreground = Brushes.Red;
104. return;
105. }
106. else
107. {
108. u\_level.Sort((a, b) => (int)a.score - (int)b.score);
109. u\_level.Reverse();
110. if (u\_level.FirstOrDefault(d => d.result == "CLEAR") == null)
111. {
112. if (u\_level.Count() == 0)
113. {
114. Player\_Name1.Text = "-";
115. Player\_Name1.Foreground = Brushes.Red;
116. Player\_from1.Text = "-";
117. Player\_from1.Foreground = Brushes.Red;
118. Player\_Score1.Text = "-";
119. Player\_Score1.Foreground = Brushes.Red;
120. Player\_Name2.Text = "-";
121. Player\_Name2.Foreground = Brushes.Red;
122. Player\_from2.Text = "-";
123. Player\_from2.Foreground = Brushes.Red;
124. Player\_Score2.Text = "-";
125. Player\_Score2.Foreground = Brushes.Red;
126. Player\_Name3.Text = "-";
127. Player\_Name3.Foreground = Brushes.Red;
128. Player\_from3.Text = "-";
129. Player\_from3.Foreground = Brushes.Red;
130. Player\_Score3.Text = "-";
131. Player\_Score3.Foreground = Brushes.Red;
132. return;
133. }
134. else
135. {
136. user\_rank first\_data = u\_level.First();
137. Player\_Name1.Text = first\_data.name;
138. Player\_Name1.Foreground = Brushes.Blue;
139. Player\_from1.Text = first\_data.from;
140. Player\_from1.Foreground = Brushes.Blue;
141. Player\_Score1.Text = first\_data.score.ToString();
142. Player\_Score1.Foreground = Brushes.Blue;
143. u\_level.Remove(first\_data);
144. }
145. }
146. else
147. {
148. user\_rank first\_data = u\_level.FirstOrDefault(d => d.result == "CLEAR");
149. Player\_Name1.Text = first\_data.name;
150. Player\_Name1.Foreground = Brushes.Black;
151. Player\_from1.Text = first\_data.from;
152. Player\_from1.Foreground = Brushes.Black;
153. Player\_Score1.Text = first\_data.score.ToString();
154. Player\_Score1.Foreground = Brushes.Black;
155. u\_level.Remove(first\_data);
156. }
157. if (u\_level.FirstOrDefault(d => d.result == "CLEAR") == null)
158. {
159. if (u\_level.Count() == 0)
160. {
161. Player\_Name2.Text = "-";
162. Player\_Name2.Foreground = Brushes.Red;
163. Player\_from2.Text = "-";
164. Player\_from2.Foreground = Brushes.Red;
165. Player\_Score2.Text = "-";
166. Player\_Score2.Foreground = Brushes.Red;
167. Player\_Name3.Text = "-";
168. Player\_Name3.Foreground = Brushes.Red;
169. Player\_from3.Text = "-";
170. Player\_from3.Foreground = Brushes.Red;
171. Player\_Score3.Text = "-";
172. Player\_Score3.Foreground = Brushes.Red;
173. return;
174. }
175. else
176. {
177. user\_rank second\_data = u\_level.First();
178. Player\_Name2.Text = second\_data.name;
179. Player\_from2.Text = second\_data.from;
180. Player\_Score2.Text = second\_data.score.ToString();
181. Player\_Name2.Foreground = Brushes.Blue;
182. Player\_from2.Foreground = Brushes.Blue;
183. Player\_Score2.Foreground = Brushes.Blue;
184. u\_level.Remove(second\_data);
185. }
186. }
187. else
188. {
189. user\_rank second\_data = u\_level.FirstOrDefault(d => d.result == "CLEAR");
190. Player\_Name2.Text = second\_data.name;
191. Player\_from2.Text = second\_data.from;
192. Player\_Score2.Text = second\_data.score.ToString();
193. Player\_Name2.Foreground = Brushes.Black;
194. Player\_from2.Foreground = Brushes.Black;
195. Player\_Score2.Foreground = Brushes.Black;
196. u\_level.Remove(second\_data);
197. }
198. if (u\_level.FirstOrDefault(d => d.result == "CLEAR") == null)
199. {
200. if (u\_level.Count() == 0)
201. {
202. Player\_Name3.Text = "-";
203. Player\_Name3.Foreground = Brushes.Red;
204. Player\_from3.Text = "-";
205. Player\_from3.Foreground = Brushes.Red;
206. Player\_Score3.Text = "-";
207. Player\_Score3.Foreground = Brushes.Red;
208. return;
209. }
210. else
211. {
212. user\_rank third\_data = u\_level.First();
213. Player\_Name3.Text = third\_data.name;
214. Player\_from3.Text = third\_data.from;
215. Player\_Score3.Text = third\_data.score.ToString();
216. Player\_Name3.Foreground = Brushes.Blue;
217. Player\_from3.Foreground = Brushes.Blue;
218. Player\_Score3.Foreground = Brushes.Blue;
219. }
220. }
221. else
222. {
223. user\_rank third\_data = u\_level.FirstOrDefault(d => d.result == "CLEAR");
224. Player\_Name3.Text = third\_data.name;
225. Player\_from3.Text = third\_data.from;
226. Player\_Score3.Text = third\_data.score.ToString();
227. Player\_Name3.Foreground = Brushes.Black;
228. Player\_from3.Foreground = Brushes.Black;
229. Player\_Score3.Foreground = Brushes.Black;
230. }
231. }
232. }
233. //HOME
234. private void Button\_Click(object sender, RoutedEventArgs e)
235. {
236. var page1 = new Page1(p\_name6, p\_from6, g\_level6);
237. NavigationService.Navigate(page1);
238. }
239. //GAME
240. private void Button\_Click1(object sender, RoutedEventArgs e)
241. {
242. var page2 = new Page2(p\_name6, p\_from6, g\_level6);
243. NavigationService.Navigate(page2);
244. }
245. //LEVEL1
246. private void Button\_Click\_lv1(object sender, RoutedEventArgs e)
247. {
248. g\_level6 = 1;
249. getRank(g\_level6);
250. }
251. //LEVEL2
252. private void Button\_Click\_lv2(object sender, RoutedEventArgs e)
253. {
254. g\_level6 = 2;
255. getRank(g\_level6);
256. }
257. //LEVEL3
258. private void Button\_Click\_lv3(object sender, RoutedEventArgs e)
259. {
260. g\_level6 = 3;
261. getRank(g\_level6);
262. }
263. //LEVEL4
264. private void Button\_Click\_lv4(object sender, RoutedEventArgs e)
265. {
266. g\_level6 = 4;
267. getRank(g\_level6);
268. }
269. //LEVEL5
270. private void Button\_Click\_lv5(object sender, RoutedEventArgs e)
271. {
272. g\_level6 = 5;
273. getRank(g\_level6);
274. }
275. //LEVEL6
276. private void Button\_Click\_lv6(object sender, RoutedEventArgs e)
277. {
278. g\_level6 = 6;
279. getRank(g\_level6);
280. }
281. //LEVEL7
282. private void Button\_Click\_lv7(object sender, RoutedEventArgs e)
283. {
284. g\_level6 = 7;
285. getRank(g\_level6);
286. }
287. }
288. }
289. 戦績画面

**<Page7.xaml>**

1. <Page x:Class="Game2.Page7"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d" d:DesignWidth="1218.201"
8. Title="Page6" Height="850" HorizontalAlignment="Center" VerticalAlignment="Center">
9. <Grid Background="White" Height="850" VerticalAlignment="Center">
10. <Label Content="BRICK BREAKER" Margin="0,10,0,0" FontSize="72" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Center" Width="668" VerticalAlignment="Top"/>
11. <Grid HorizontalAlignment="Center" Margin="0,288,0,0" Width="938" VerticalAlignment="Top">
12. <Label Content="Score" Margin="258,0,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Left" Width="214" VerticalAlignment="Top"/>
13. <Label Content="Result" Margin="532,0,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Left" Width="400" VerticalAlignment="Top"/>
14. <Label Content="Rank" HorizontalAlignment="Left" Margin="0,0,0,0" VerticalAlignment="Top" Height="78" Width="194" FontSize="60" FontFamily="Broadway"/>
15. <Label Content="No1:" HorizontalAlignment="Left" Margin="0,80,0,0" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
16. <Label Content="No2:" HorizontalAlignment="Left" Margin="0,160,0,0" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
17. <Label Content="No3:" HorizontalAlignment="Left" Margin="0,240,0,0" VerticalAlignment="Top" Height="78" Width="229" FontSize="60" FontFamily="Broadway"/>
18. <TextBox x:Name="Player\_Score1" HorizontalAlignment="Left" Height="78" Margin="202,80,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
19. <TextBox x:Name="Player\_res1" HorizontalAlignment="Left" Height="78" Margin="532,80,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="400" SelectionBrush="#FF71B9F2" FontSize="48"/>
20. <TextBox x:Name="Player\_Score2" HorizontalAlignment="Left" Height="78" Margin="202,160,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
21. <TextBox x:Name="Player\_res2" HorizontalAlignment="Left" Height="78" Margin="532,160,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="400" SelectionBrush="#FF71B9F2" FontSize="48"/>
22. <TextBox x:Name="Player\_Score3" HorizontalAlignment="Left" Height="78" Margin="202,240,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="327" SelectionBrush="#FF71B9F2" FontSize="48"/>
23. <TextBox x:Name="Player\_res3" HorizontalAlignment="Left" Height="78" Margin="532,240,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="400" SelectionBrush="#FF71B9F2" FontSize="48"/>
24. </Grid>
25. <Grid Margin="0,700,0,0" HorizontalAlignment="Center" VerticalAlignment="Top" Width="710">
26. <Button Content="GAME" HorizontalAlignment="Left" Height="120" VerticalAlignment="Top" Width="230" Click="Button\_Click1"/>
27. <Button Content="HOME" Height="120" Margin="240,0,0,0" VerticalAlignment="Top" Click="Button\_Click" HorizontalAlignment="Left" Width="230"/>
28. <Button Content="LOG OUT" HorizontalAlignment="Left" Height="120" Margin="480,0,0,0" VerticalAlignment="Top" Width="230" Click="Button\_Click1"/>
29. </Grid>
30. <Label Content="STATES" Margin="0,101,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Center" Width="256" VerticalAlignment="Top"/>
31. <Grid Height="62" Margin="0,220,0,0" Width="406" HorizontalAlignment="Center" VerticalAlignment="Top">
32. <TextBox x:Name="Player\_level" TextWrapping="Wrap" FontFamily="Broadway" FontSize="48" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="206,0,0,0" Height="60"/>
33. <Label Content="Level:" FontSize="48" Background="{x:Null}" FontFamily="Broadway" Margin="0,0,222,0" Height="60"/>
34. </Grid>
35. <Grid Margin="0,629,0,0" Width="770" Height="61" HorizontalAlignment="Center" VerticalAlignment="Top">
36. <Button Content="LEVEL1" Click="Button\_Click\_lv1" Margin="0,0,680,0"/>
37. <Button Content="LEVEL2" Click="Button\_Click\_lv2" Margin="113,0,565,0"/>
38. <Button Content="LEVEL3" Click="Button\_Click\_lv3" Margin="226,0,450,0"/>
39. <Button Content="LEVEL4" Click="Button\_Click\_lv4" Margin="339,0,340,0"/>
40. <Button Content="LEVEL5" Click="Button\_Click\_lv5" Margin="452,0,226,0"/>
41. <Button Content="LEVEL6" Click="Button\_Click\_lv6" Margin="566,0,114,0"/>
42. <Button Content="LEVEL7" Click="Button\_Click\_lv7" Margin="680,0,0,0"/>
43. </Grid>
44. <Grid Height="60" Margin="0,170,0,0" Width="831" HorizontalAlignment="Center" VerticalAlignment="Top">
45. <TextBox x:Name="Player\_ID" TextWrapping="Wrap" FontFamily="Broadway" FontSize="48" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="206,0,425,0" Height="60"/>
46. <Label Content="ID:" FontSize="48" Background="{x:Null}" FontFamily="Broadway" Margin="0,0,647,0" Height="60"/>
47. <TextBox x:Name="Player\_Name" TextWrapping="Wrap" FontFamily="Broadway" FontSize="48" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="631,0,0,0" Height="60"/>
48. <Label Content="Name:" FontSize="48" Background="{x:Null}" FontFamily="Broadway" Margin="425,0,222,0" Height="60"/>
49. </Grid>
50. </Grid>
51. </Page>

**<Page7.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.IO;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. using System.Windows;
8. using System.Windows.Controls;
9. using System.Windows.Data;
10. using System.Windows.Documents;
11. using System.Windows.Input;
12. using System.Windows.Media;
13. using System.Windows.Media.Imaging;
14. using System.Windows.Navigation;
15. using System.Windows.Shapes;
16. namespace Game2
17. {
18. /// <summary>
19. /// Page6.xaml の相互作用ロジック
20. /// </summary>
21. public partial class Page7 : Page
22. {
23. double g\_level7 = 1;
24. string p\_name7;
25. string p\_from7;
26. string p\_id7;
27. List<Page4.user> U4 = new List<Page4.user>();
28. public Page7(string p\_name, string p\_from, double level)
29. {
30. InitializeComponent();
31. p\_name7 = p\_name;
32. Player\_Name.Text = p\_name7;
33. Player\_ID.IsReadOnly = true;
34. Player\_Name.IsReadOnly = true;
35. Player\_level.IsReadOnly = true;
36. Player\_Score1.IsReadOnly = true;
37. Player\_Score2.IsReadOnly = true;
38. Player\_Score3.IsReadOnly = true;
39. Player\_res1.IsReadOnly = true;
40. Player\_res2.IsReadOnly = true;
41. Player\_res3.IsReadOnly = true;
42. g\_level7 = level;
43. p\_from7 = p\_name;
44. getRank(g\_level7);
45. }
46. public void getRank(double level)
47. {
48. Player\_level.Text = level.ToString();
49. List<Page6.user\_rank> u6 = new List<Page6.user\_rank>();
50. var filePath\_U = "..\\test.csv";
51. // 読み込みたいCSVファイルのパスを指定して開く
52. StreamReader sr\_U = new StreamReader(filePath\_U);
53. {
54. // 末尾まで繰り返す
55. while (!sr\_U.EndOfStream)
56. {
57. // CSVファイルの一行を読み込む
58. string line = sr\_U.ReadLine();
59. // 読み込んだ一行をカンマ毎に分けて配列に格納する
60. string[] values = line.Split(',');
61. Page4.user nu = new Page4.user();
62. nu.id = values[0];
63. nu.name = values[1];
64. nu.pass = values[2];
65. nu.from = values[3];
66. nu.score = Double.Parse(values[4]);
67. U4.Add(nu);
68. }
69. }
70. p\_id7 = U4.Find(u => u.name == p\_name7).id;
71. Player\_ID.Text = p\_id7;
72. var filePath = "..\\result.csv";
73. // 読み込みたいCSVファイルのパスを指定して開く
74. StreamReader sr = new StreamReader(filePath);
75. {
76. // 末尾まで繰り返す
77. while (!sr.EndOfStream)
78. {
79. // CSVファイルの一行を読み込む
80. string line = sr.ReadLine();
81. // 読み込んだ一行をカンマ毎に分けて配列に格納する
82. string[] values = line.Split(',');
83. Page6.user\_rank nur = new Page6.user\_rank();
84. nur.name = values[0];
85. nur.from = values[1];
86. nur.level = Double.Parse(values[2]);
87. nur.score = Double.Parse(values[3]);
88. nur.result = values[4];
89. u6.Add(nur);
90. }
91. }
92. List<Page6.user\_rank> u\_level = u6.FindAll(u => u.level == level);
93. u\_level.Sort((a, b) => (int)a.score - (int)b.score);
94. u\_level.Reverse();
95. if (u\_level.FirstOrDefault(d => d.result == "CLEAR") == null)
96. {
97. if (u\_level.Count() == 0)
98. {
99. Player\_res1.Text = "-";
100. Player\_res1.Foreground = Brushes.Red;
101. Player\_Score1.Text = "-";
102. Player\_Score1.Foreground = Brushes.Red;
103. Player\_res2.Text = "-";
104. Player\_res2.Foreground = Brushes.Red;
105. Player\_Score2.Text = "-";
106. Player\_Score2.Foreground = Brushes.Red;
107. Player\_res3.Text = "-";
108. Player\_res3.Foreground = Brushes.Red;
109. Player\_Score3.Text = "-";
110. Player\_Score3.Foreground = Brushes.Red;
111. return;
112. }
113. else
114. {
115. Page6.user\_rank first\_data = u\_level.First();
116. Player\_res1.Text = first\_data.result;
117. Player\_res1.Foreground = Brushes.Blue;
118. Player\_Score1.Text = first\_data.score.ToString();
119. Player\_Score1.Foreground = Brushes.Blue;
120. u\_level.Remove(first\_data);
121. }
122. }
123. else
124. {
125. Page6.user\_rank first\_data = u\_level.FirstOrDefault(d => d.result == "CLEAR");
126. Player\_res1.Text = first\_data.result;
127. Player\_res1.Foreground = Brushes.Black;
128. Player\_Score1.Text = first\_data.score.ToString();
129. Player\_Score1.Foreground = Brushes.Black;
130. u\_level.Remove(first\_data);
131. }
132. if (u\_level.FirstOrDefault(d => d.result == "CLEAR") == null)
133. {
134. if (u\_level.Count() == 0)
135. {
136. Player\_res2.Text = "-";
137. Player\_res2.Foreground = Brushes.Red;
138. Player\_Score2.Text = "-";
139. Player\_Score2.Foreground = Brushes.Red;
140. Player\_res3.Text = "-";
141. Player\_res3.Foreground = Brushes.Red;
142. Player\_Score3.Text = "-";
143. Player\_Score3.Foreground = Brushes.Red;
144. return;
145. }
146. else
147. {
148. Page6.user\_rank second\_data = u\_level.First();
149. Player\_res2.Text = second\_data.result;
150. Player\_Score2.Text = second\_data.score.ToString();
151. Player\_res2.Foreground = Brushes.Blue;
152. Player\_Score2.Foreground = Brushes.Blue;
153. u\_level.Remove(second\_data);
154. }
155. }
156. else
157. {
158. Page6.user\_rank second\_data = u\_level.FirstOrDefault(d => d.result == "CLEAR");
159. Player\_res2.Text = second\_data.result;
160. Player\_Score2.Text = second\_data.score.ToString();
161. Player\_res2.Foreground = Brushes.Black;
162. Player\_Score2.Foreground = Brushes.Black;
163. u\_level.Remove(second\_data);
164. }
165. if (u\_level.FirstOrDefault(d => d.result == "CLEAR") == null)
166. {
167. if (u\_level.Count() == 0)
168. {
169. Player\_res3.Text = "-";
170. Player\_res3.Foreground = Brushes.Red;
171. Player\_Score3.Text = "-";
172. Player\_Score3.Foreground = Brushes.Red;
173. return;
174. }
175. else
176. {
177. Page6.user\_rank third\_data = u\_level.First();
178. Player\_res3.Text = third\_data.result;
179. Player\_Score3.Text = third\_data.score.ToString();
180. Player\_res3.Foreground = Brushes.Blue;
181. Player\_Score3.Foreground = Brushes.Blue;
182. }
183. }
184. else
185. {
186. Page6.user\_rank third\_data = u\_level.FirstOrDefault(d => d.result == "CLEAR");
187. Player\_res3.Text = third\_data.result;
188. Player\_Score3.Text = third\_data.score.ToString();
189. Player\_res3.Foreground = Brushes.Black;
190. Player\_Score3.Foreground = Brushes.Black;
191. }
192. }
193. //HOME
194. private void Button\_Click(object sender, RoutedEventArgs e)
195. {
196. var page1 = new Page1(p\_name7, p\_from7, g\_level7);
197. NavigationService.Navigate(page1);
198. }
199. //GAME
200. private void Button\_Click1(object sender, RoutedEventArgs e)
201. {
202. var page2 = new Page2(p\_name7, p\_from7, g\_level7);
203. NavigationService.Navigate(page2);
204. }
205. //LEVEL1
206. private void Button\_Click\_lv1(object sender, RoutedEventArgs e)
207. {
208. g\_level7 = 1;
209. getRank(g\_level7);
210. }
211. //LEVEL2
212. private void Button\_Click\_lv2(object sender, RoutedEventArgs e)
213. {
214. g\_level7 = 2;
215. getRank(g\_level7);
216. }
217. //LEVEL3
218. private void Button\_Click\_lv3(object sender, RoutedEventArgs e)
219. {
220. g\_level7 = 3;
221. getRank(g\_level7);
222. }
223. //LEVEL4
224. private void Button\_Click\_lv4(object sender, RoutedEventArgs e)
225. {
226. g\_level7 = 4;
227. getRank(g\_level7);
228. }
229. //LEVEL5
230. private void Button\_Click\_lv5(object sender, RoutedEventArgs e)
231. {
232. g\_level7 = 5;
233. getRank(g\_level7);
234. }
235. //LEVEL6
236. private void Button\_Click\_lv6(object sender, RoutedEventArgs e)
237. {
238. g\_level7 = 6;
239. getRank(g\_level7);
240. }
241. //LEVEL7
242. private void Button\_Click\_lv7(object sender, RoutedEventArgs e)
243. {
244. g\_level7 = 7;
245. getRank(g\_level7);
246. }
247. }
248. }
249. 画面/ゲーム説明画面

**<Page8.xaml>**

1. <Page x:Class="Game2.Page8"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d" d:DesignWidth="1218.201"
8. Title="Page6" Height="850" HorizontalAlignment="Center" VerticalAlignment="Center">
9. <Grid Background="White" Height="850" VerticalAlignment="Center">
10. <Label Content="BRICK BREAKER" Margin="0,10,0,0" FontSize="72" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Center" Width="668" VerticalAlignment="Top"/>
11. <Grid HorizontalAlignment="Center" Margin="0,170,0,0" Width="938" VerticalAlignment="Top" Height="495">
12. <Label Content="EXPLANATION:" Margin="400,0,0,0" FontSize="60" FontWeight="Bold" FontFamily="Broadway" HorizontalAlignment="Left" Width="520" VerticalAlignment="Top"/>
13. <Label Content="UI:" HorizontalAlignment="Left" Margin="0,0,0,0" VerticalAlignment="Top" Height="78" Width="194" FontSize="60" FontFamily="Broadway"/>
14. <Image x:Name="UIimage" HorizontalAlignment="Left" Width="400" Margin="0,80,0,0" Stretch="UniformToFill"/>
15. <TextBox x:Name="explanation" HorizontalAlignment="Left" Height="415" Margin="400,80,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="538" SelectionBrush="#FF71B9F2" FontSize="18"/>
16. </Grid>
17. <Grid Margin="0,740,0,0" HorizontalAlignment="Center" VerticalAlignment="Top" Width="500">
18. <Button Content="GAME" HorizontalAlignment="Right" Height="80" VerticalAlignment="Top" Width="230" Click="Button\_Click1"/>
19. <Button Content="HOME" Height="80" Margin="0,0,0,0" VerticalAlignment="Top" Click="Button\_Click" HorizontalAlignment="Left" Width="230"/>
20. </Grid>
21. <Grid Height="80" Margin="0,101,0,0" Width="700" HorizontalAlignment="Center" VerticalAlignment="Top">
22. <TextBox x:Name="Manual" TextWrapping="Wrap" FontFamily="Broadway" FontSize="60" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="313,0,0,10" Height="80" VerticalAlignment="Center"/>
23. <Label Content="MANUAL:" FontSize="60" Background="{x:Null}" FontFamily="Broadway" Margin="0,0,83,0" Height="80" HorizontalAlignment="Left" VerticalAlignment="Center"/>
24. </Grid>
25. <Grid Margin="0,670,0,0" Width="770" Height="61" HorizontalAlignment="Center" VerticalAlignment="Top">
26. <Button Content="LOG IN" Click="Button\_Click\_lv1" Margin="0,0,680,0"/>
27. <Button Content="REGISTRATION" Click="Button\_Click\_lv2" Margin="113,0,565,0"/>
28. <Button Content="HOME" Click="Button\_Click\_lv3" Margin="226,0,450,0"/>
29. <Button Content="GAME" Click="Button\_Click\_lv4" Margin="339,0,340,0"/>
30. <Button Content="RANKING" Click="Button\_Click\_lv5" Margin="452,0,226,0"/>
31. <Button Content="RESULT" Click="Button\_Click\_lv6" Margin="566,0,114,0"/>
32. <Button Content="STATES" Click="Button\_Click\_lv7" Margin="680,0,0,0"/>
33. </Grid>
34. </Grid>
35. </Page>

**<Page8.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.Drawing;
4. using System.IO;
5. using System.Linq;
6. using System.Reflection;
7. using System.Text;
8. using System.Threading.Tasks;
9. using System.Windows;
10. using System.Windows.Controls;
11. using System.Windows.Data;
12. using System.Windows.Documents;
13. using System.Windows.Input;
14. using System.Windows.Media;
15. using System.Windows.Media.Imaging;
16. using System.Windows.Navigation;
17. using System.Windows.Shapes;
18. namespace Game2
19. {
20. /// <summary>
21. /// Page6.xaml の相互作用ロジック
22. /// </summary>
23. public partial class Page8 : Page
24. {
25. string manual = "LOG IN";
26. string p\_name8;
27. string p\_from8;
28. double p\_level8;
29. List<Page4.user> U4 = new List<Page4.user>();
30. public Page8(string p\_name, string p\_from, double p\_level)
31. {
32. InitializeComponent();
33. p\_level8 = p\_level;
34. p\_from8 = p\_from;
35. p\_name8 = p\_name;
36. explanation.IsReadOnly = true;
38. setEx(manual);
39. }
40. public void setEx(string manual)
41. {
42. Manual.Text = manual;
44. string ImagePath = checkPath(manual);
45. UIimage.Source = new BitmapImage(new Uri(ImagePath, UriKind.Relative));
46. UIimage.Stretch = Stretch.Uniform;
47. explanation.Text = checkEx(manual);
48. }
49. public string checkEx(string manual)
50. {
51. string str = "";
52. switch (manual)
53. {
54. case "LOG IN":
55. str =
56. "< Screen Description >\n" +
57. " If you are registered as a user, \n" +
58. " please enter your ID and Pass and log in.\n" +
59. "< Button >\n" +
60. " [ LOG IN ]\n" +
61. " After entering your ID and Pass,\n" +
62. " authenticate the user.\n" +
63. " [ NEW PLAYER ]\n" +
64. " Go to the user registration screen.\n" +
65. " [ GUEST PLAY ]\n" +
66. " Log in as a guest.\n" +
67. " Some functions are not available.\n";
68. break;
69. case "REGISTRATION":
70. str =
71. "< Screen Description >\n" +
72. " Register as a user.\n" +
73. "< Button >\n" +
74. " [ LOG IN ]\n" +
75. " Go to the log in screen.\n" +
76. " [ REGISTRATION ]\n" +
77. " Check the information you entered and register the user.\n" +
78. " [ ALL CLEAR ]\n" +
79. " Delete all the information you enter.\n";
80. break;
81. case "HOME":
82. str =
83. "< Screen Description >\n" +
84. " You can use various features from home.\n" +
85. "< Button >\n" +
86. " [ GAME ]\n" +
87. " Go to the game screen.\n" +
88. " [ RANKING ]\n" +
89. " Go to the ranking screen.\n" +
90. " [ STATES ]\n" +
91. " Go to the stats screen.\n" +
92. " [ MANUAL ]\n" +
93. " Go to the manual screen.\n" +
94. " [ LOG OUT ]\n" +
95. " Go to the log in screen.\n";
96. break;
97. case "GAME":
98. str =
99. "< Screen Description >\n" +
100. " It is a play screen.\n" +
101. " You can choose a level and play.\n" +
102. " Move the bar to operate the red ball\n" +
103. " and erase the purple block displayed\n" +
104. " on the play screen after the game starts.\n" +
105. " It will be clear if all blocks can be erased.\n" +
106. " The bar at the bottom of the gameplay screen\n" +
107. " can be moved to the left with the 1 key\n" +
108. " and to the right with the 0 key.\n" +
109. "< Button >\n" +
110. " [ SATRT ]\n" +
111. " Start the game.\n" +
112. " [ STOP ]\n" +
113. " Pause the game.\n" +
114. " [ RESATRT ]\n" +
115. " Resume the game.\n" +
116. " [ RESET ]\n" +
117. " Reset the game.\n" +
118. " [ RANKING ]\n" +
119. " Go to the ranking screen.\n" +
120. " If you move to another screen,\n" +
121. " the game data will not be recorded.\n" +
122. " [ AUTO ]\n" +
123. " The game is played automatically.\n" +
124. " If you use auto, the score will be halved.\n" +
125. " It does not always clear.\n" +
126. " [ HOME ]\n" +
127. " Go to the home screen.\n" +
128. " If you move to another screen,\n" +
129. " the game data will not be recorded.\n" +
130. " [ LEVEL1 ] ~ [ LEVEL7 ]\n" +
131. " You can select the level of the game.";
132. break;
133. case "RESULT":
134. str =
135. "< Screen Description >\n" +
136. " Display game results.\n" +
137. "< Button >\n" +
138. " [ FORWARD ]\n" +
139. " Lower the game level by one level.\n" +
140. " [ REGAME ]\n" +
141. " You can try the same level game again.\n" +
142. " [ NEXT ]\n" +
143. " Raise the level of the game by one level.\n" +
144. " [ RANKING ]\n" +
145. " Go to the ranking screen.\n" +
146. " If you move to another screen,\n" +
147. " you can't go back to the result screen.\n" +
148. " [ HOME ]\n" +
149. " Go to the home screen.\n" +
150. " If you move to another screen,\n" +
151. " you can't go back to the result screen.\n";
152. break;
153. case "RANKING":
154. str =
155. "< Screen Description >\n" +
156. " The ranking is displayed.\n" +
157. " Those who have cleared are displayed in black,\n" +
158. " and those who have not cleared are displayed in blue.\n" +
159. "< Button >\n" +
160. " [ HOME ]\n" +
161. " Go to the home screen.\n" +
162. " [ GAME ]\n" +
163. " Go to the game screen.\n" +
164. " [ LEVEL1 ] ~ [ LEVEL7 ]\n" +
165. " Shows the ranking of the selected game level.";
166. break;
167. case "STATES":
168. str =
169. "< Screen Description >\n" +
170. " Show your game record.\n" +
171. "< Button >\n" +
172. " [ HOME ]\n" +
173. " Go to the home screen.\n" +
174. " [ GAME ]\n" +
175. " Go to the game screen.\n" +
176. " [ LOG OUT ]\n" +
177. " Go to the log in screen.\n"+
178. " [ LEVEL1 ] ~ [ LEVEL7 ]\n" +
179. " Shows the your record of the selected game level.";
180. break;
181. }
182. return str;
183. }
184. public string checkPath(string manual)
185. {
186. string ImagePath = "";
187. switch (manual)
188. {
189. case "LOG IN":
190. ImagePath = "NewFolder1/Page4.png";
191. break;
192. case "REGISTRATION":
193. ImagePath = "NewFolder1/Page5.png";
194. break;
195. case "HOME":
196. ImagePath = "NewFolder1/Page1.png";
197. break;
198. case "GAME":
199. ImagePath = "NewFolder1/Page2.png";
200. break;
201. case "RESULT":
202. ImagePath = "NewFolder1/Page3.png";
203. break;
204. case "RANKING":
205. ImagePath = "NewFolder1/Page6.png";
206. break;
207. case "STATES":
208. ImagePath = "NewFolder1/Page7.png";
209. break;
210. }
211. return ImagePath;
212. }
213. //HOME
214. private void Button\_Click(object sender, RoutedEventArgs e)
215. {
216. var page1 = new Page1(p\_name8, p\_from8, p\_level8);
217. NavigationService.Navigate(page1);
218. }
219. //GAME
220. private void Button\_Click1(object sender, RoutedEventArgs e)
221. {
222. var page2 = new Page2(p\_name8, p\_from8, p\_level8);
223. NavigationService.Navigate(page2);
224. }
225. //LOG IN
226. private void Button\_Click\_lv1(object sender, RoutedEventArgs e)
227. {
228. setEx(manual);
229. manual = "LOG IN";
230. }
231. //REGISTRATION
232. private void Button\_Click\_lv2(object sender, RoutedEventArgs e)
233. {
234. manual = "REGISTRATION";
235. setEx(manual);
236. }
237. //HOME
238. private void Button\_Click\_lv3(object sender, RoutedEventArgs e)
239. {
240. manual = "HOME";
241. setEx(manual);
242. }
243. //GAME
244. private void Button\_Click\_lv4(object sender, RoutedEventArgs e)
245. {
246. manual = "GAME";
247. setEx(manual);
248. }
249. //RESULT
250. private void Button\_Click\_lv5(object sender, RoutedEventArgs e)
251. {
252. manual = "RESULT";
253. setEx(manual);
254. }
255. //RANKING
256. private void Button\_Click\_lv6(object sender, RoutedEventArgs e)
257. {
258. manual = "RANKING";
259. setEx(manual);
260. }
261. //STATES
262. private void Button\_Click\_lv7(object sender, RoutedEventArgs e)
263. {
264. manual = "STATES";
265. setEx(manual);
266. }
267. }
268. }
269. ゲーム画面

**<Page2.xaml>**

**<Page2.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. using System.Timers;
7. using System.Windows;
8. using System.Windows.Controls;
9. using System.Windows.Data;
10. using System.Windows.Documents;
11. using System.Windows.Input;
12. using System.Windows.Media;
13. using System.Windows.Media.Imaging;
14. using System.Windows.Navigation;
15. using System.Windows.Shapes;
16. namespace Game2
17. {
18. /// <summary>
19. /// Page1.xaml の相互作用ロジック
20. /// </summary>
21. public partial class Page2: Page
22. {
23. private Timer timer;
24. Key? leftOrRightKey = null;
25. List<Ball> balls = new List<Ball>();
26. List<Block> blocks = new List<Block>();
27. Paddle paddle;
28. double count;
29. double c = 0;
30. double btn\_AUTO = 2;
31. double level = 1;
32. double column; //ブロックの列数
33. double row;
34. double score = 0;
35. double s\_game = 0;
36. string res = "";
37. string P\_From = "";
38. public Page2(string p\_Name, string p\_From, double g\_level)
39. {
40. InitializeComponent();
41. Player\_Info.IsReadOnly = true;
42. Player\_Level.IsReadOnly = true;
43. Player\_Score.IsReadOnly = true;
44. if(p\_Name == "")
45. {
46. p\_Name = "Guest";
47. }
48. if(p\_From == "")
49. {
50. p\_From = "-";
51. }
52. Player\_Info.Text = p\_Name;
53. P\_From = p\_From;
54. Player\_Score.Text = score.ToString();
55. level = g\_level;
56. Player\_Level.Text = level.ToString();
57. }
58. public void gameSet()
59. {
60. score = 0;
61. Player\_Score.Text = score.ToString();
62. Player\_Level.Text = level.ToString();
63. c = 0;
64. //パドル
65. paddle = new Paddle(stage, 100, 10, stage.Width / 2 - 50, stage.Height - 50, 4 + level );
66. stage.Children.Add(paddle.getShape());
67. //ボールの作成
68. double l = 18;
69. Ball ball = new Ball(stage, l, l, stage.Width / 2 - l / 2, stage.Height - 50 - l);
70. balls.Add(ball);
71. stage.Children.Add(ball.getShape());
72. double blockWidth;
73. double blockHeight;
74. double w, s;
75. //ブロックの作成
76. if (level == 1)
77. {
78. blockWidth = 100; //ブロックの横幅
79. blockHeight = 40; //ブロックの縦幅
80. column = 4; //ブロックの列数
81. row = 3; //ブロックの行数
82. w = stage.Width;
83. s = (w - column \* blockWidth) / (column + 1);
84. count = column \* row;
85. for (int i = 0; i < row; i++)
86. {
87. for (int j = 0; j < column; j++)
88. {
89. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
90. blocks.Add(block);
91. stage.Children.Add(block.getShape());
92. }
93. }
94. }
95. else if(level== 2)
96. {
97. blockWidth = 80; //ブロックの横幅
98. blockHeight = 24; //ブロックの縦幅
99. column = 5; //ブロックの列数
100. row = 5; //ブロックの行数
101. w = stage.Width;
102. s = (w - column \* blockWidth) / (column + 1);
103. for (int i = 0; i < row; i++)
104. {
105. for (int j = 0; j < column; j++)
106. {
107. if(i % 2 == 0)
108. {
109. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
110. blocks.Add(block);
111. stage.Children.Add(block.getShape());
112. count = count + 1;
113. }
114. }
115. }
116. }
117. else if (level== 3)
118. {
119. blockWidth = 400 / 6; //ブロックの横幅
120. blockHeight = 24; //ブロックの縦幅
121. column = 6; //ブロックの列数
122. row = 5; //ブロックの行数
123. w = stage.Width;
124. s = (w - column \* blockWidth) / (column + 1);
125. for (int i = 0; i < row; i++)
126. {
127. for (int j = 0; j < column; j++)
128. {
129. if (i % 2 == 0)
130. {
131. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
132. blocks.Add(block);
133. stage.Children.Add(block.getShape());
134. count = count + 1;
135. }
136. else
137. {
138. if (j % 2 != 0)
139. {
140. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
141. blocks.Add(block);
142. stage.Children.Add(block.getShape());
143. count = count + 1;
144. }
145. }
146. }
147. }
148. }
149. else if (level == 4)
150. {
151. blockWidth = 400 / 7; //ブロックの横幅
152. blockHeight = 20; //ブロックの縦幅
153. column = 7; //ブロックの列数
154. row = 6; //ブロックの行数
155. w = stage.Width;
156. s = (w - column \* blockWidth) / (column + 1);
157. for (int i = 0; i < row; i++)
158. {
159. for (int j = 0; j < column; j++)
160. {
161. if (i % 2 == 0)
162. {
163. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
164. blocks.Add(block);
165. stage.Children.Add(block.getShape());
166. count = count + 1;
167. }
168. else
169. {
170. if(j % 3 == 0)
171. {
172. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
173. blocks.Add(block);
174. stage.Children.Add(block.getShape());
175. count = count + 1;
176. }
177. }
178. }
179. }
180. }
181. else if (level == 5)
182. {
183. blockWidth = 50; //ブロックの横幅
184. blockHeight = 120/7; //ブロックの縦幅
185. column = 8; //ブロックの列数
186. row = 7; //ブロックの行数
187. w = stage.Width;
188. s = (w - column \* blockWidth) / (column + 1);
189. for (int i = 0; i < row; i++)
190. {
191. for (int j = 0; j < column; j++)
192. {
193. if (i % 3 == 0)
194. {
195. if (j % 2 == 0)
196. {
197. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
198. blocks.Add(block);
199. stage.Children.Add(block.getShape());
200. count = count + 1;
201. }
202. }
203. else if (i % 3 == 1)
204. {
205. if (j % 3 != 0)
206. {
207. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
208. blocks.Add(block);
209. stage.Children.Add(block.getShape());
210. count = count + 1;
211. }
212. }
213. else
214. {
215. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
216. blocks.Add(block);
217. stage.Children.Add(block.getShape());
218. count = count + 1;
219. }
220. }
221. }
222. }
223. else if (level == 6)
224. {
225. blockWidth = 400 / 15; //ブロックの横幅
226. blockHeight = 120 / 9; //ブロックの縦幅
227. column = 15; //ブロックの列数
228. row = 9; //ブロックの行数
229. w = stage.Width;
230. s = (w - column \* blockWidth) / (column + 1);
231. for (int i = 0; i < row; i++)
232. {
233. for (int j = 0; j < column; j++)
234. {
235. if (i % 4 == 1)
236. {
237. if (j % 4 == 0)
238. {
239. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
240. blocks.Add(block);
241. stage.Children.Add(block.getShape());
242. count = count + 1;
243. }
244. }
245. else if (i % 4 == 2)
246. {
247. if (j % 2 != 0)
248. {
249. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
250. blocks.Add(block);
251. stage.Children.Add(block.getShape());
252. count = count + 1;
253. }
254. }
255. else if (i % 4 == 3)
256. {
257. if (j % 3 != 0)
258. {
259. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
260. blocks.Add(block);
261. stage.Children.Add(block.getShape());
262. count = count + 1;
263. }
264. }
265. else
266. {
267. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
268. blocks.Add(block);
269. stage.Children.Add(block.getShape());
270. count = count + 1;
271. }
272. }
273. }
274. }
275. else if (level == 7)
276. {
277. blockWidth = 20; //ブロックの横幅
278. blockHeight = 12; //ブロックの縦幅
279. column = 20; //ブロックの列数
280. row = 10; //ブロックの行数
281. w = stage.Width;
282. s = (w - column \* blockWidth) / (column + 1);
283. count = column \* row;
284. for (int i = 0; i < row; i++)
285. {
286. for (int j = 0; j < column; j++)
287. {
288. if (i % 4 == 1)
289. {
290. if (j % 5 != 0)
291. {
292. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
293. blocks.Add(block);
294. stage.Children.Add(block.getShape());
295. count = count + 1;
296. }
297. }
298. else if (i % 4 == 2)
299. {
300. if (j % 3 == 0)
301. {
302. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
303. blocks.Add(block);
304. stage.Children.Add(block.getShape());
305. count = count + 1;
306. }
307. }
308. else if(i % 4 == 3)
309. {
310. if (j % 7 != 0)
311. {
312. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
313. blocks.Add(block);
314. stage.Children.Add(block.getShape());
315. count = count + 1;
316. }
317. }
318. else
319. {
320. if (j % 2 != 0)
321. {
322. Block block = new Block(stage, blockWidth, blockHeight, (s + blockWidth) \* j + s, 50 + i \* (blockHeight + 10));
323. blocks.Add(block);
324. stage.Children.Add(block.getShape());
325. count = count + 1;
326. }
327. }
328. }
329. }
330. }
331. //タイマー
332. timer = new Timer(1.0 / 60);
333. timer.Elapsed += updateDisplay;
334. timer.Start();
335. }
336. public void updateDisplay(object sender, ElapsedEventArgs e)
337. {
338. try
339. {
340. Dispatcher.Invoke(() =>
341. {
342. paddle.nextFrame(leftOrRightKey);
343. balls.ForEach(ball =>
344. {
345. score = get\_Score(ball, btn\_AUTO);
346. if (ball.getBottom() > stage.Height)
347. {
348. timer.Stop();
349. res = "GAME OVER";
350. var page3 = new Page3(Player\_Info.Text, P\_From, level, get\_Score(ball, btn\_AUTO), res);
351. NavigationService.Navigate(page3);
352. }
353. if (btn\_AUTO % 2 != 0)
354. {
355. if ((ball.getLeft() + ball.getRight()) / 2 < (paddle.getLeft() + paddle.getRight()) / 2)
356. {
357. leftOrRightKey = Key.D1;
358. }
359. else if ((ball.getLeft() + ball.getRight()) / 2 > (paddle.getLeft() + paddle.getRight()) / 2)
360. {
361. leftOrRightKey = Key.D0;
362. }
363. }
364. Player\_Score.Text = score.ToString();
365. //ボールを進める
366. ball.nextFrame();
367. //ブロックの当たり判定
368. blocks.ForEach(block =>
369. {
370. string re = collisionBlock(block, ball);
371. if (re != null)
372. {
373. c = c + 1;
374. stage.Children.Remove(block.getShape());
375. blocks.Remove(block);
376. ball.reflect(re, ball.getDx());
377. }
378. if (c == count)
379. {
380. timer.Stop();
381. res = "CLEAR";
382. var page3 = new Page3(Player\_Info.Text, P\_From, level, get\_Score(ball, btn\_AUTO), res);
383. NavigationService.Navigate(page3);
384. }
385. });
386. //パネルとの当たり判定
387. bool b = collosionPaddle(paddle, ball);
388. if (b)
389. {
390. double dx = ball.getDx();
391. if(leftOrRightKey == Key.D1)
392. {
393. if(dx >= 0)
394. {
395. dx = dx - (level + 4) \* 0.05;
396. }
397. else if(dx < 0)
398. {
399. dx = dx + (level + 4) \* 0.05;
400. }
401. }
402. else if(leftOrRightKey == Key.D0)
403. {
404. if (dx >= 0)
405. {
406. dx = dx + (level + 4) \* 0.05;
407. }
408. else if (dx < 0)
409. {
410. dx = dx - (level + 4) \* 0.05;
411. }
412. }
413. ball.reflect("by", dx);
414. }
415. });
416. });
417. }
418. catch (TaskCanceledException)
419. {
420. Console.WriteLine("Error!!");
421. }
422. }
423. protected void Windouws\_KeyDown(object sender, KeyEventArgs e)
424. {
425. if (btn\_AUTO % 2 == 1)
426. {
427. return;
428. }
429. if (e.Key == Key.D1|| e.Key == Key.D0)
430. {
431. leftOrRightKey = e.Key;
432. }
433. }
434. protected void Windouws\_KeyUp(object sender, KeyEventArgs e)
435. {
436. if (btn\_AUTO % 2 == 1)
437. {
438. return;
439. }
440. if (e.Key == Key.D1 || e.Key == Key.D0)
441. {
442. leftOrRightKey = null;
443. }
444. }
445. //ブロックとボールの当たり判定
446. public static string collisionBlock(Block block, Ball ball)
447. {
448. if (ball.getTop() > block.getBottom() ||
449. ball.getBottom() < block.getTop() ||
450. ball.getLeft() > block.getRight() ||
451. ball.getRight() < block.getLeft())
452. {
453. return null;
454. }
455. //当たり方
456. if (ball.getRight() < block.getRight() && ball.getLeft() > block.getBottom())
457. {
458. return "y";
459. }
460. else if (ball.getTop() > block.getTop() && ball.getBottom() < block.getBottom())
461. {
462. return "x";
463. }
464. else
465. {
466. return "xy";
467. }
468. }
469. //パドルの当たり判定
470. public static bool collosionPaddle(Paddle paddle, Ball ball)
471. {
472. //当たらない判定
473. if (ball.getTop() > paddle.getBottom() ||
474. ball.getBottom() < paddle.getTop() ||
475. ball.getLeft() > paddle.getRight() ||
476. ball.getRight() < paddle.getLeft())
477. {
478. return false;
479. }
480. return true;
481. }
482. //スコア計算
483. public static double get\_Score(Ball ball, double auto)
484. {
485. if(ball.getS\_kabe() - ball.getS\_block() > 0)
486. {
487. if(auto != 2)
488. {
489. return (ball.getS\_block() + (ball.getS\_kabe() - ball.getS\_block()) \* 10)/2;
490. }
491. return ball.getS\_block() + (ball.getS\_kabe() - ball.getS\_block()) \* 10;
492. }
493. else
494. {
495. if (auto != 2)
496. {
497. return ball.getS\_block() / 2;
498. }
499. return ball.getS\_block();
500. }
501. }
502. //RESETボタン
503. private void Button\_Click(object sender, RoutedEventArgs e)
504. {
505. timer.Stop();
506. stage.Children.Remove(paddle.getShape());
507. ballsClear();
508. blocksClear();
509. }
510. public void ballsClear()
511. {
512. balls.ForEach(ball =>
513. {
514. if (ball.getShape() != null)
515. {
516. stage.Children.Remove(ball.getShape());
517. }
518. });
519. balls.RemoveAll(ball => ball.getShape() != null);
520. }
521. public void blocksClear()
522. {
523. blocks.ForEach(block =>
524. {
525. if (block.getShape() != null)
526. {
527. stage.Children.Remove(block.getShape());
528. }
529. });
530. blocks.RemoveAll(block => block.getShape() != null);
531. }
532. //STOP
533. private void Button\_Click\_1(object sender, RoutedEventArgs e)
534. {
535. timer.Stop();
536. }
537. //RESTART
538. private void Button\_Click\_2(object sender, RoutedEventArgs e)
539. {
540. if(c >= 12)
541. {
542. return;
543. }
544. timer.Start();
545. }
546. //AUTO
547. private void Button\_Click\_3(object sender, RoutedEventArgs e)
548. {
549. btn\_AUTO++;
550. }
551. //HOME
552. private void Button\_Click\_HOME(object sender, RoutedEventArgs e)
553. {
554. if(s\_game != 0)
555. {
556. timer.Stop();
557. }
558. var page1 = new Page1(Player\_Info.Text,P\_From, level);
559. NavigationService.Navigate(page1);
560. }
561. //START
562. private void Button\_Ckick\_START(object sender, RoutedEventArgs e)
563. {
564. if (s\_game != 0)
565. {
566. timer.Stop();
567. stage.Children.Remove(paddle.getShape());
568. ballsClear();
569. blocksClear();
570. }
571. s\_game = s\_game + 1;
572. gameSet();
573. }
574. //RANKING
575. private void Button\_Ckick\_RANKING(object sender, RoutedEventArgs e)
576. {
577. if (s\_game != 0)
578. {
579. timer.Stop();
580. }
581. var page6 = new Page6(Player\_Info.Text, P\_From, 1);
582. NavigationService.Navigate(page6);
583. }
584. //LEVEL1
585. private void Button\_Click\_lv1(object sender, RoutedEventArgs e)
586. {
587. if (s\_game != 0)
588. {
589. timer.Stop();
590. stage.Children.Remove(paddle.getShape());
591. ballsClear();
592. blocksClear();
593. }
594. level = 1;
595. Player\_Level.Text = level.ToString();
596. }
597. //LEVEL2
598. private void Button\_Click\_lv2(object sender, RoutedEventArgs e)
599. {
600. if (s\_game != 0)
601. {
602. timer.Stop();
603. stage.Children.Remove(paddle.getShape());
604. ballsClear();
605. blocksClear();
606. }
607. level = 2;
608. Player\_Level.Text = level.ToString();
609. }
610. //LEVEL3
611. private void Button\_Click\_lv3(object sender, RoutedEventArgs e)
612. {
613. if (s\_game != 0)
614. {
615. timer.Stop();
616. stage.Children.Remove(paddle.getShape());
617. ballsClear();
618. blocksClear();
619. }
620. level = 3;
621. Player\_Level.Text = level.ToString();
622. }
623. //LEVEL4
624. private void Button\_Click\_lv4(object sender, RoutedEventArgs e)
625. {
626. if (s\_game != 0)
627. {
628. timer.Stop();
629. stage.Children.Remove(paddle.getShape());
630. ballsClear();
631. blocksClear();
632. }
633. level = 4;
634. Player\_Level.Text = level.ToString();
635. }
636. //LEVEL5
637. private void Button\_Click\_lv5(object sender, RoutedEventArgs e)
638. {
639. if (s\_game != 0)
640. {
641. timer.Stop();
642. stage.Children.Remove(paddle.getShape());
643. ballsClear();
644. blocksClear();
645. }
646. level = 5;
647. Player\_Level.Text = level.ToString();
648. }
649. //LEVEL6
650. private void Button\_Click\_lv6(object sender, RoutedEventArgs e)
651. {
652. if (s\_game != 0)
653. {
654. timer.Stop();
655. stage.Children.Remove(paddle.getShape());
656. ballsClear();
657. blocksClear();
658. }
659. level = 6;
660. Player\_Level.Text = level.ToString();
661. }
662. //LEVEL7
663. private void Button\_Click\_lv7(object sender, RoutedEventArgs e)
664. {
665. if (s\_game != 0)
666. {
667. timer.Stop();
668. stage.Children.Remove(paddle.getShape());
669. ballsClear();
670. blocksClear();
671. }
672. level = 7;
673. Player\_Level.Text = level.ToString();
674. }
675. }
676. public class Block
677. {
678. Canvas stage;
679. Rectangle shape = new Rectangle();
680. double x, y, dx, dy;
681. public Block(Canvas stage, double width = 20, double height = 20,
682. double x = 0, double y = 0,
683. double dx = 2, double dy = -2, Brush brush = null)
684. {
685. this.stage = stage;
686. shape.Width = width;
687. shape.Height = height;
688. this.x = x;
689. this.y = y;
690. this.dx = dx;
691. this.dy = dy;
692. if (brush != null)
693. {
694. shape.Stroke = brush;
695. }
696. else
697. {
698. shape.Fill = Brushes.BlueViolet;
699. }
700. Canvas.SetLeft(shape, x);
701. Canvas.SetTop(shape, y);
702. }
703. public Rectangle getShape()
704. {
705. return shape;
706. }
707. public double getTop()
708. {
709. return y;
710. }
711. public double getLeft()
712. {
713. return x;
714. }
715. public double getBottom()
716. {
717. return y + shape.Height;
718. }
719. public double getRight()
720. {
721. return x + shape.Width;
722. }
723. }
724. public class Paddle
725. {
726. Canvas stage;
727. Rectangle shape = new Rectangle();
728. double x, y, dx;
729. public Paddle(Canvas stage, double width = 200, double height = 10,
730. double x = 10, double y = 0, double dx = 5, Brush brush = null)
731. {
732. this.stage = stage;
733. shape.Width = width;
734. shape.Height = height;
735. this.x = x;
736. this.y = y;
737. this.dx = dx;
738. if (brush != null)
739. {
740. shape.Stroke = brush;
741. }
742. else
743. {
744. shape.Stroke = Brushes.Black;
745. shape.Fill = Brushes.Black;
746. }
747. Canvas.SetLeft(shape, x);
748. Canvas.SetTop(shape, y);
749. }
750. public Rectangle getShape()
751. {
752. return shape;
753. }
754. public double getTop()
755. {
756. return y;
757. }
758. public double getLeft()
759. {
760. return x;
761. }
762. public double getBottom()
763. {
764. return y + shape.Height;
765. }
766. public double getRight()
767. {
768. return x + shape.Width;
769. }
770. public void nextFrame(Key? key)
771. {
772. if (key == null)
773. {
774. return;
775. }
776. else if (key == Key.D1)
777. {
778. x -= dx;
779. if (x < 0)
780. {
781. x = 0;
782. }
783. }
784. else if (key == Key.D0)
785. {
786. x += dx;
787. if (x + shape.Width > stage.Width)
788. {
789. x = stage.Width - shape.Width;
790. }
791. }
792. Canvas.SetLeft(shape, x);
793. }
794. }
795. public class Ball
796. {
797. Canvas stage;
798. Ellipse shape = new Ellipse();
799. double x, y, dx, dy, ddx, ddy, s\_kabe = 0, s\_block = 0, s\_hantei = 0;
800. public Ball(Canvas stage, double width = 20, double height = 20,
801. double x = 10, double y = 0,
802. double dx = 2, double dy = -2,
803. double ddx = 0.2, double ddy = -0.2, Brush brush = null)
804. {
805. this.stage = stage;
806. shape.Width = width;
807. shape.Height = height;
808. this.x = x;
809. this.y = y;
810. this.dx = dx;
811. this.dy = dy;
812. this.ddx = ddx;
813. this.ddy = ddy;
814. if (brush != null)
815. {
816. shape.Stroke = brush;
817. }
818. else
819. {
820. shape.Fill = Brushes.Red;
821. }
822. Canvas.SetLeft(shape, x);
823. Canvas.SetTop(shape, y);
824. }
825. public Ellipse getShape()
826. {
827. return shape;
828. }
829. public double getTop()
830. {
831. return y;
832. }
833. public double getLeft()
834. {
835. return x;
836. }
837. public double getBottom()
838. {
839. return y + shape.Height;
840. }
841. public double getRight()
842. {
843. return x + shape.Width;
844. }
845. public void reflect(string s, double d)
846. {
847. this.dx = d;
848. if (s == "x")
849. {
850. s\_hantei++;
851. s\_block = s\_block + s\_hantei \* s\_hantei;
852. dx \*= -1;
853. }
854. else if(s == "by")
855. {
856. s\_hantei++;
857. s\_block = s\_block + 2 \* s\_hantei;
858. dy \*= -1;
859. }
860. else if (s == "y")
861. {
862. s\_hantei++;
863. s\_block = s\_block + 2 \* s\_hantei;
864. dy \*= -1;
865. }
866. else
867. {
868. s\_hantei++;
869. s\_block = s\_block + 2 \* s\_hantei;
870. dx \*= -1;
871. dy \*= -1;
872. }
873. }
874. public void accelerrateBall(string s)
875. {
876. if (s == "x")
877. {
878. if (dx > 0)
879. {
880. dx += ddx;
881. }
882. else
883. {
884. dx -= ddx;
885. }
886. }
887. else if (s == "y")
888. {
889. if (dx > 0)
890. {
891. dx += ddx;
892. }
893. else
894. {
895. dx -= ddx;
896. }
897. }
898. }
899. public void nextFrame()
900. {
901. x += dx;
902. y += dy;
903. if (getRight() >= stage.Width)
904. {
905. s\_hantei = 0;
906. s\_kabe++;
907. x = stage.Width - (getRight() - stage.Width) - shape.Width;
908. dx \*= -1;
909. accelerrateBall("x");
910. }
911. else if (x <= 0)
912. {
913. s\_hantei = 0;
914. s\_kabe++;
915. x \*= -1;
916. dx \*= -1;
917. accelerrateBall("x");
918. }
919. else if (y <= 0)
920. {
921. s\_hantei = 0;
922. s\_kabe++;
923. y \*= -1;
924. dy \*= -1;
925. accelerrateBall("y");
926. }
927. Canvas.SetLeft(shape, x);
928. Canvas.SetTop(shape, y);
929. }
930. public double getS\_kabe()
931. {
932. return s\_kabe;
933. }
934. public double getS\_block()
935. {
936. return s\_block;
937. }
938. public double getS\_hantei()
939. {
940. return s\_hantei + 1;
941. }
942. public double getDx()
943. {
944. return this.dx;
945. }
946. }
947. }
948. リザルト画面

**<Page3.xaml>**

1. <Page x:Class="Game2.Page3"
2. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6. xmlns:local="clr-namespace:Game2"
7. mc:Ignorable="d"
8. d:DesignHeight="450" d:DesignWidth="800"
9. Title="Page3" Height="450" Width="800" HorizontalAlignment="Center" VerticalAlignment="Center">
10. <Grid>
11. <Canvas Background="White" Margin="0,0,0,0">
12. <Grid Canvas.Left="56" Canvas.Top="320" Width="690" Height="100" HorizontalAlignment="Center">
13. <Button Content="REGAME" Width="130" Click="Button\_Click" Margin="140,0,0,0" HorizontalAlignment="Left"/>
14. <Button Content="RANKING" Width="130" Click="Button\_Ckick\_RANKING" Margin="420,0,0,0" HorizontalAlignment="Left"/>
15. <Button Content="NEXT" Width="130" Click="Button\_Ckick\_NEXT" HorizontalAlignment="Center"/>
16. <Button Content="HOME" Width="130" Click="Button\_Click\_HOME" Margin="0,0,0,0" HorizontalAlignment="Right"/>
17. <Button Content="FORWARD" Width="130" Click="Button\_Ckick\_Forward" Margin="0,0,0,0" HorizontalAlignment="Left"/>
18. </Grid>
19. <Grid Height="131" Canvas.Left="115" Canvas.Top="170" Width="515" HorizontalAlignment="Center">
20. <TextBox x:Name="Player\_Score" TextWrapping="Wrap" FontFamily="Broadway" FontSize="36" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="215,80,0,0"/>
21. <TextBox x:Name="Player\_Info" TextWrapping="Wrap" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" FontSize="36" FontFamily="Broadway" Margin="215,0,0,80"/>
22. <Label Content="Player:" Width="200" Margin="0,0,0,80" FontSize="36" Background="{x:Null}" FontFamily="Broadway" HorizontalAlignment="Left" />
23. <Label Content="Score:" Width="200" Margin="0,80,0,0" FontSize="36" Background="{x:Null}" FontFamily="Broadway" HorizontalAlignment="Left" />
24. <TextBox x:Name="Player\_level" TextWrapping="Wrap" FontFamily="Broadway" FontSize="36" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" Margin="215,40,0,40"/>
25. <Label Content="Level:" Width="200" Margin="0,40,0,40" FontSize="36" FontFamily="Broadway" HorizontalAlignment="Left" />
26. </Grid>
27. <TextBox x:Name="Player\_Result" TextWrapping="Wrap" Background="{x:Null}" BorderBrush="{x:Null}" SelectionBrush="{x:Null}" FontSize="72" FontFamily="Broadway" Width="620" Canvas.Top="37" Height="118" FontWeight="Bold" TextAlignment="Center" Canvas.Left="91" HorizontalAlignment="Center"/>
28. </Canvas>
29. </Grid>
30. </Page>

**<Page3.xaml.cs>**

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. using System.Windows;
7. using System.Windows.Controls;
8. using System.Windows.Data;
9. using System.Windows.Documents;
10. using System.Windows.Input;
11. using System.Windows.Media;
12. using System.Windows.Media.Imaging;
13. using System.Windows.Navigation;
14. using System.Windows.Shapes;
15. namespace Game2
16. {
17. /// <summary>
18. /// Page3.xaml の相互作用ロジック
19. /// </summary>
20. public partial class Page3 : Page
21. {
22. string P\_From = "";
23. double level = 0;
24. public Page3(string P\_name, string P\_from, double P\_level, double P\_score, string P\_res)
25. {
26. InitializeComponent();
27. Player\_Info.Text = P\_name;
28. Player\_Score.Text = P\_score.ToString();
29. Player\_level.Text = P\_level.ToString();
30. Player\_Result.Text = P\_res;
31. Player\_Info.IsReadOnly = true;
32. Player\_level.IsReadOnly = true;
33. Player\_Score.IsReadOnly = true;
34. Player\_Result.IsReadOnly = true;
35. P\_From = P\_from;
36. level = P\_level;
37. var filePath2 = "..\\result.csv";
38. string[] values = { Player\_Info.Text, P\_From, level.ToString(), Player\_Score.Text, Player\_Result.Text };
39. using (var sw = new System.IO.StreamWriter(filePath2, true))
40. {
41. sw.WriteLine("{0},{1},{2},{3},{4}", values[0], values[1], values[2], values[3], values[4]);
42. }
43. }
44. //Home
45. private void Button\_Click\_HOME(object sender, RoutedEventArgs e)
46. {
47. var page1 = new Page1(Player\_Info.Text, P\_From, level);
48. NavigationService.Navigate(page1);
49. }
50. //Game
51. private void Button\_Click(object sender, RoutedEventArgs e)
52. {
53. var page2 = new Page2(Player\_Info.Text, P\_From, level);
54. NavigationService.Navigate(page2);
55. }
56. //Ranking
57. private void Button\_Ckick\_RANKING(object sender, RoutedEventArgs e)
58. {
59. var page6 = new Page6(Player\_Info.Text, P\_From, 1);
60. NavigationService.Navigate(page6);
61. }
62. //NEXT
63. private void Button\_Ckick\_NEXT(object sender, RoutedEventArgs e)
64. {
65. if(Player\_Result.Text != "CLEAR")
66. {
67. MessageBox.Show("Not clear. \nPlease clear and come back.");
68. return;
69. }
70. if(level != 7)
71. {
72. var page2 = new Page2(Player\_Info.Text, P\_From, level + 1);
73. NavigationService.Navigate(page2);
74. }
75. else
76. {
77. var page2 = new Page2(Player\_Info.Text, P\_From, 1);
78. NavigationService.Navigate(page2);
79. }
81. }
82. //FORWARD
83. private void Button\_Ckick\_Forward(object sender, RoutedEventArgs e)
84. {
85. if(level != 1)
86. {
87. var page2 = new Page2(Player\_Info.Text, P\_From, level - 1);
88. NavigationService.Navigate(page2);
89. }
90. else
91. {
92. var page2 = new Page2(Player\_Info.Text, P\_From, 7);
93. NavigationService.Navigate(page2);
94. }
95. }
96. }
97. }

* **参考にしたWebサイトなど**

【C# TextBox】入力を禁止する (1)

<[【C# TextBox】入力を禁止する (1) - ITLAB51.COM](https://www.itlab51.com/?p=2759)> (最終閲覧：2021/1/31)

C# でリストが空かどうかを確認する

<[C# でリストが空かどうかを確認する | Delft スタック (delftstack.com)](https://www.delftstack.com/ja/howto/csharp/check-if-list-is-empty-in-csharp/#c%23-%25E3%2581%25AE-list.any-%25E9%2596%25A2%25E6%2595%25B0%25E3%2582%2592%25E4%25BD%25BF%25E7%2594%25A8%25E3%2581%2597%25E3%2581%25A6%25E3%2583%25AA%25E3%2582%25B9%25E3%2583%2588%25E3%2581%258C%25E7%25A9%25BA%25E3%2581%258B%25E3%2581%25A9%25E3%2581%2586%25E3%2581%258B%25E3%2582%2592%25E7%25A2%25BA%25E8%25AA%258D%25E3%2581%2599%25E3%2582%258B)> (最終閲覧：2021/1/31)

イメージングの概要

< [イメージングの概要 - WPF .NET Framework | Microsoft Docs](https://docs.microsoft.com/ja-jp/dotnet/desktop/wpf/graphics-multimedia/imaging-overview?view=netframeworkdesktop-4.8)> (最終閲覧：2021/1/31)

C#で画像を読み込む方法とは？様々な方法をご紹介！

< [C#で画像を読み込む方法とは？様々な方法をご紹介！ | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/tool/3850/)> (最終閲覧：2021/1/31)

WPFで画像を表示する

< [WPFで画像を表示する (vdlz.xyz)](https://www.vdlz.xyz/Csharp/WPF/Control/Image/Image.html)> (最終閲覧：2021/1/31)

【C#】画像ファイルを指定したサイズ（幅×高さ）に変更する

< [【C#】画像ファイルを指定したサイズ（幅×高さ）に変更する - ktts’s blog (hatenablog.com)](https://ktts.hatenablog.com/entry/2018/10/15/004708)> (最終閲覧：2021/1/31)

CSSで縦横比を維持しながら横幅いっぱいに背景画像を表示させる方法

< [CSSで縦横比を維持しながら横幅いっぱいに背景画像を表示させる方法 - NxWorld](https://www.nxworld.net/css-background-aspect-ratio.html)> (最終閲覧：2021/1/31)

【C#】WPFで画像表示する2つの方法【初心者向け】

< [【C#】WPFで画像表示する2つの方法【初心者向け】 | すぽてく (shinshin-log.com)](https://shinshin-log.com/csharp-insertimage/)> (最終閲覧：2021/1/31)

【C#・LINQ】Anyメソッドの概要と使い方について

< [【C#・LINQ】Anyメソッドの概要と使い方について | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/language/2162/)> (最終閲覧：2021/1/31)

【C#入門】Listの要素を検索する方法総まとめ(Find/IndexOf/Contains)

< [【C#入門】Listの要素を検索する方法総まとめ(Find/IndexOf/Contains) | 侍エンジニアブログ (sejuku.net)](https://www.sejuku.net/blog/45381)> (最終閲覧：2021/1/31)

C＃コードでWPFテキストボックスの背景色を設定する

< [C＃コードでWPFテキストボックスの背景色を設定する (qastack.jp)](https://qastack.jp/programming/979876/set-background-color-of-wpf-textbox-in-c-sharp-code)> (最終閲覧：2021/1/31)

【C#】テキストボックスの背景色・文字色を変更する

< [【C#】テキストボックスの背景色・文字色を変更する (stellacreate.com)](https://stellacreate.com/entry/cs_textbox_color_change)> (最終閲覧：2021/1/31)

【C#入門】Listの要素を検索するFindの使い方(FindAll/FindIndex)

< [【C#入門】Listの要素を検索するFindの使い方(FindAll/FindIndex) | 侍エンジニアブログ (sejuku.net)](https://www.sejuku.net/blog/45252)> (最終閲覧：2021/1/31)

【C#入門】Listの使い方総まとめ(ArrayList/Add/Remove/ソート/検索)

< [【C#入門】Listの使い方総まとめ(ArrayList/Add/Remove/ソート/検索) | 侍エンジニアブログ (sejuku.net)](https://www.sejuku.net/blog/47378)> (最終閲覧：2021/1/31)

【C#】【WPF】画面を最大化！全画面表示する方法

< [【C#】【WPF】画面を最大化！全画面表示する方法 | Step1 (alfort.online)](https://alfort.online/2137)> (最終閲覧：2021/1/31)

C#を使って配列やリストから要素を検索する方法

< [C#を使って配列やリストから要素を検索する方法 | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/language/5367/#%E3%83%AA%E3%82%B9%E3%83%88%E3%81%8B%E3%82%89%E8%A6%81%E7%B4%A0%E3%82%92%E6%A4%9C%E7%B4%A2%E3%81%97%E3%80%81%E5%8F%96%E5%BE%97%E3%81%99%E3%82%8B%E6%96%B9%E6%B3%95)> (最終閲覧：2021/1/31)

【C#入門】IndexOfで文字列やList、配列を検索する方法

< [【C#入門】IndexOfで文字列やList、配列を検索する方法 | 侍エンジニアブログ (sejuku.net)](https://www.sejuku.net/blog/41736)> (最終閲覧：2021/1/31)

C#のListから要素を削除する方法について紹介します！使い方を覚えて活用しよう

< [C#のListから要素を削除する方法について紹介します！使い方を覚えて活用しよう | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/language/c-sharp/8206/#Remove%E3%81%A7%E5%89%8A%E9%99%A4)> (最終閲覧：2021/1/31)

[C# LINQ] 配列、リストから条件を満たす最初の要素を取得する（First、FirstOrDefault）

< [[C# LINQ] 配列、リストから条件を満たす最初の要素を取得する（First、FirstOrDefault） (yaspage.com)](https://yaspage.com/prog/csharp/cs-linq-first/#toc3)> (最終閲覧：2021/1/31)

LINQ：先頭のデータを取得する - Firstメソッド［C#］

< [LINQ：先頭のデータを取得する - Firstメソッド［C#］ - Build Insider](https://www.buildinsider.net/web/bookaspmvc5/050307)> (最終閲覧：2021/1/31)

Enumerable.​First メソッド / List< T>.First メソッドを利用する - 最初の要素を取得する (C#プログラミング)

< [Enumerable.​First メソッド / List< T>.First メソッドを利用する - 最初の要素を取得する : C# | iPentec](https://www.ipentec.com/document/csharp-enumerable-first-metod)> (最終閲覧：2021/1/31)

C#でテキストファイルやCSVファイルを読み込む方法

< [C#でテキストファイルやCSVファイルを読み込む方法 | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/%E8%A8%80%E8%AA%9E%E3%83%BB%E7%92%B0%E5%A2%83/1226/#C%E3%81%A7CSV%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%82%92%E8%AA%AD%E3%81%BF%E8%BE%BC%E3%82%80%E6%96%B9%E6%B3%95)> (最終閲覧：2021/1/31)

【C#の基礎】Array.Sortを使った配列のソート方法

< [【C#の基礎】Array.Sortを使った配列のソート方法 | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/%e8%a8%80%e8%aa%9e%e3%83%bb%e7%92%b0%e5%a2%83/1463/)> (最終閲覧：2021/1/31)

(C#)Listで要素を検索し取得する

< [(C#)Listで要素を検索し取得する - Qiita](https://qiita.com/watame/items/9602b8d2506005b08157)> (最終閲覧：2021/1/31)

C#のリストボックスの作成・選択・削除・追加の方法とは？

< [C#のリストボックスの作成・選択・削除・追加の方法とは？ | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/language/3717/)> (最終閲覧：2021/1/31)

【C#-文法】List多次元配列の値追加と出力

< [【C#-文法】List多次元配列の値追加と出力 - 学習の型 (hatenablog.com)](https://gaishiengineer.hatenablog.com/entry/2020/08/21/230857)> (最終閲覧：2021/1/31)

[C#] 2次元配列のようにListを使用する方法（List）

< [[C#] 2次元配列のようにListを使用する方法（List）｜初心者エンジニアのための備忘録 (turtle-engineers.com)](https://turtle-engineers.com/c-niji-list/)> (最終閲覧：2021/1/31)

List< T> から配列へ変換する (C#プログラミング)

< [List< T> から配列へ変換する : C# | iPentec](https://www.ipentec.com/document/csharp-convert-generics-list-to-array)> (最終閲覧：2021/1/31)

クラスを実装する (C#プログラミング)

< [クラスを実装する : C# | iPentec](https://www.ipentec.com/document/csharp-implement-class)> (最終閲覧：2021/1/31)

【コピペで使える】超簡単！C#でCSV書き込み

< [【コピペで使える】超簡単！C#でCSV書き込み | 趣味や仕事に役立つ初心者DIYプログラミング入門 (resanaplaza.com)](https://resanaplaza.com/2021/10/02/%e3%80%90%e3%82%b3%e3%83%94%e3%83%9a%e3%81%a7%e4%bd%bf%e3%81%88%e3%82%8b%e3%80%91%e8%b6%85%e7%b0%a1%e5%8d%98%ef%bc%81c%e3%81%a7csv%e6%9b%b8%e3%81%8d%e8%be%bc%e3%81%bf/#StreamWriter)> (最終閲覧：2021/1/31)

【C#】もう悩まない！CSV読み込みの最も簡単な方法を徹底解説

< [【C#】もう悩まない！CSV読み込みの最も簡単な方法を徹底解説 | 侍エンジニアブログ (sejuku.net)](https://www.sejuku.net/blog/85579)> (最終閲覧：2021/1/31)

Listの要素を検索するには？［C#／VB］

< [Listの要素を検索するには？［C#／VB］：.NET TIPS - ＠IT (itmedia.co.jp)](https://atmarkit.itmedia.co.jp/ait/articles/1703/15/news019.html)> (最終閲覧：2021/1/31)

[C#] Listをコピーする

< [[C#] Listをコピーする - C#ちょこっとリファレンス (programmer-reference.com)](https://csharp.programmer-reference.com/list-copy/)> (最終閲覧：2021/1/31)

C# の 2次元配列みたいな List のソート

< [C# の 2次元配列みたいな List のソート - 解析エンジニアの自動化 blog (hatenablog.com)](https://chuckischarles.hatenablog.com/entry/2019/07/01/101727)> (最終閲覧：2021/1/31)

２次元配列をCSVで書き出し（UTF8形式）VBA

< [２次元配列をCSVで書き出し（UTF8形式）VBA - Qiita](https://qiita.com/takeru0x5569/items/50737d4478d8071bbb5c)> (最終閲覧：2021/1/31)

DataTableや配列等をCSV形式のファイルとして保存する

< [DataTableや配列等をCSV形式のファイルとして保存する - .NET Tips (VB.NET,C#...) (dobon.net)](https://dobon.net/vb/dotnet/file/writecsvfile.html)> (最終閲覧：2021/1/31)

[C# プログラミング入門] 配列について

< [[C# 入門] 配列の初期化・宣言・使い方 (yaspage.com)](https://yaspage.com/prog/csharp/cs-array/#:~:text=%E9%85%8D%E5%88%97%E3%81%AE%E5%AE%A3%E8%A8%80%E3%81%AF%E3%81%93%E3%82%93%E3%81%AA%E6%84%9F%E3%81%98%E3%81%A7%E3%81%99%E3%80%82%20%E5%9E%8B%E3%81%AE%E5%BE%8C%E3%82%8D%E3%81%AB%20%E8%A7%92%E3%81%8B%E3%81%A3%E3%81%93%20%5B%20%5D%20%E3%82%92%E4%BB%98%E3%81%91%E3%82%8B%E3%81%A8%E9%85%8D%E5%88%97%E3%81%AB%E3%81%AA%E3%82%8A%E3%81%BE%E3%81%99%E3%80%82%20%E5%9E%8B%20%5B%5D,int%20%5B%5D%20a%3B%20%2F%2Fint%E5%9E%8B%E3%81%AE%E9%85%8D%E5%88%97%20string%20%5B%5D%20b%3B%20%2F%2Fstring%E5%9E%8B%E3%81%AE%E9%85%8D%E5%88%97)> (最終閲覧：2021/1/31)

[C#]リストをカンマ区切りの文字列で出力する

< [[C#]リストをカンマ区切りの文字列で出力する (ht-jp.net)](http://ht-jp.net/blog/pc/c/c-string-join)> (最終閲覧：2021/1/31)

C# で CSVファイル を List データ に 読み込む 方法

< [C# で CSVファイル を List データ に 読み込む 方法 - galife (garafu.blogspot.com)](https://garafu.blogspot.com/2014/10/c-csv-list.html)> (最終閲覧：2021/1/31)

C#でCSVファイルにデータを書き込む

< [C#でCSVファイルにデータを書き込む - whoopsidaisies's diary (hatenablog.com)](https://whoopsidaisies.hatenablog.com/entry/2013/12/03/141352)> (最終閲覧：2021/1/31)

[C#][.NET] CSVファイルの書き出し(列とプロパティをバインディング、属性を使って)

< [[C#][.NET] CSVファイルの書き出し(列とプロパティをバインディング、属性を使って) - Qiita](https://qiita.com/sengoku/items/a0ce1c543aefb0fa29cb)> (最終閲覧：2021/1/31)

C# Attributeを利用した、お手軽！CSVの入出力クラスを作ろう！

< [C# Attributeを利用した、お手軽！CSVの入出力クラスを作ろう！ - Qiita](https://qiita.com/messhi/items/0f28512087e9cc71e24a)> (最終閲覧：2021/1/31)

C#でのCSVファイルの出力とは？方法をご紹介！

< [C#でのCSVファイルの出力とは？方法をご紹介！ | .NETコラム (fenet.jp)](https://www.fenet.jp/dotnet/column/language/3790/)> (最終閲覧：2021/1/31)

C# List<T>の2次元配列List<List<T>>

< [C# List<T>の2次元配列List<List<T>> (vdlz.xyz)](http://vdlz.xyz/Csharp/Csharp/List/2dList.html)> (最終閲覧：2021/1/31)

C#で、utf8(BOM無し)のテキストファイルを出力する

< [C#で、utf8(BOM無し)のテキストファイルを出力する - Qiita](https://qiita.com/tricogimmick/items/c605d1d9c56e7d582bbd)> (最終閲覧：2021/1/31)

C#で二次元配列（リスト）を扱う

< [C#で二次元配列（リスト）を扱う - Qiita](https://qiita.com/mamoru_moriai/items/8551a3807cd5c585f54e)> (最終閲覧：2021/1/31)

Listから重複した要素を削除するには？［C#／VB］

< [Listから重複した要素を削除するには？［C#／VB］：.NET TIPS - ＠IT (itmedia.co.jp)](https://atmarkit.itmedia.co.jp/ait/articles/1703/29/news027.html)> (最終閲覧：2021/1/31)

[C#]テキスト/CSV/Excelのファイルを読み取る方法

< [[C#]テキスト/CSV/Excelのファイルを読み取る方法 - Qiita](https://qiita.com/HKK_note/items/88d4f26e41a76dc92ea9)> (最終閲覧：2021/1/31)

文字列に特定の文字列が含まれているかを調べるには？（Contains編）［C#／VB、.NET 2.0］

< [文字列に特定の文字列が含まれているかを調べるには？（Contains編）［C#／VB、.NET 2.0］：.NET TIPS - ＠IT (itmedia.co.jp)](https://atmarkit.itmedia.co.jp/ait/articles/0602/17/news119.html)> (最終閲覧：2021/1/31)

Listに要素を追加／挿入するには？［C#／VB］

< [Listに要素を追加／挿入するには？［C#／VB］：.NET TIPS - ＠IT (itmedia.co.jp)](https://atmarkit.itmedia.co.jp/ait/articles/1703/01/news042.html)> (最終閲覧：2021/1/31)

[C# File] CSVファイルの読み込みと書き込み< [[C# File] CSVファイルの読み込みと書き込み (yaspage.com)](https://yaspage.com/prog/csharp/cs-csvfile/)> (最終閲覧：2021/1/31)

Windows GUIプログラミング入門6 画面遷移(2)

< [Windows GUIプログラミング入門6 画面遷移(2) - Qiita](https://qiita.com/Kosen-amai/items/9f5e56e8c80b348b127f)> (最終閲覧：2021/1/31)

キー操作

<[キー操作 - C#プチリファレンス (csharp-ref.com)](https://csharp-ref.com/program_keyoperation.html)>(最終閲覧：2021/1/31)

相対パスから絶対パスを取得する、絶対パスから相対パスを取得する

<[相対パスから絶対パスを取得する、絶対パスから相対パスを取得する - .NET Tips (VB.NET,C#...) (dobon.net)](https://dobon.net/vb/dotnet/file/getabsolutepath.html#section1)> (最終閲覧：2021/1/31)