DTT
APPS WEB GAMES

SINCE 2010

Level selection

Documentation | 21-07-22

Table of Contents

1. Get started quickly

To implement the level select in your own game quickly you can follow this workflow:

1. Create a new class that inherits from LevelSelectHandler.
    - Replace Tconfig, Tresult, Tminigame by your configuration object, your result object, your game manager object.

```
/// <summary>
/// Handle the level selection.
/// </summary>
Gael Courmont  IL code
public class CalculationRaceLevelSelectHandler : LevelSelectHandler<GameSettings, GameResult, CalculationRaceManager>
```

    - Override GetConfig (give a configuration object to return). see example below:

```
/// <summary>
/// The _levelConfigs field contains the configurable levels for each level for level selection.
/// </summary>
[SerializeField]
private BubbleShooterConfig[] _levelConfigs;  Serializable

/// <summary>
/// <inheritdoc/>
/// </summary>
/// <param name="levelNumber"><inheritdoc/></param>
/// <returns><inheritdoc/></returns>
protected override BubbleShooterConfig GetConfig(int levelNumber) => _levelConfigs[levelNumber];
```

- Override CalculateScore to return a float between 0 and 1.

```
/// <summary>
/// Calculate te score asigned to the game.
/// </summary>
/// <param name="result">Result of the game.</param>
/// <returns>The score, float between 0 and 1.</returns>
protected override float CalculateScore(GameResult result)
{
    // Number of point according to time spend to complete the game.
    // 3 point if time<30sec, 2 point if 30sec<time<60sec,
    // 1 point if 60sec<time<90sec, 0 point if time>90sec
    int Point = 3 - Mathf.clamp( (int) (result.timeTaken / 30, 0, 3);

    // return the score point in a 0 to 1 range
    // 0<point<3 -> 0<point<1
    // see Mathf.Lerp doc for more info
    return Mathf.Lerp(0, 3, point);
}
```

- example of a full implementation :

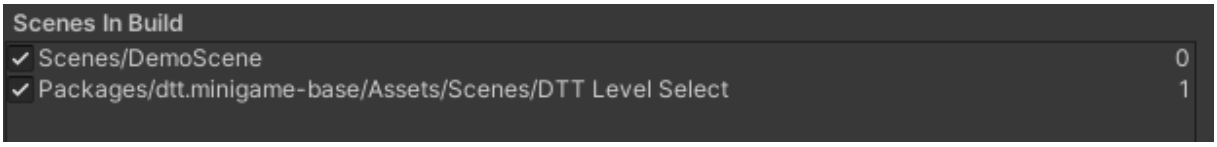
```
public class CalculationRaceLevelSelectHandler : LevelSelectHandler<GameSettings, GameResult, CalculationRaceManager>
{
    /// <summary>
    /// List of all level in the game.
    /// </summary>
    [SerializeField]
    [Tooltip("Different level of the game.")]
    private List<GameSettings> gameLevel;

    /// <summary>
    /// Get the config of the current level.
    /// </summary>
    /// <param name="levelNumber">The index of the current level</param>
    /// <returns>Settings of the chossed level</returns>
    protected override GameSettings GetConfig(int levelNumber)
    {
        return gameLevel[levelNumber % gameLevel.Count];
    }

    /// <summary>
    /// Calculate te score asigned to the game.
    /// </summary>
    /// <param name="result">Result of the game.</param>
    /// <returns>The score, float between 0 and 1.</returns>
    protected override float CalculateScore(GameResult result)
    {
        // Number of point according to time spend to complete the game.
        // 3 point if time<30sec, 2 point if 30sec<time<60sec,
        // 1 point if 60sec<time<90sec, 0 point if time>90sec
        int Point = 3 - Mathf.clamp( (int) (result.timeTaken / 30, 0, 3);

        // return the score point in a 0 to 1 range
        // 0<point<3 -> 0<point<1
        // see Mathf.Lerp doc for more info
        return Mathf.Lerp(0, 3, point);
    }
}
```

2. Add your newly created LevelSelectHandler object to your scene on any active object.
3. In your project folder, create a new LevelDatabase ScriptableObject (under“Create/DTT/Minigame Base/Level Database“)
4. Add a reference to your Level Database object in the Level SelectHandler (refer to created in step 1 and 2) of your scene.
5. Add the dtt level select scene to your build. Make sure it is not the first scene in order.



6. You can enter playmode and your level select scene should now work.

2. Introduction

The level select package is a system for handling level selection that can be reused for each game. That package allows you to easily add a custom level selection to your game that will store game result data and lock/unlock status of levels.

Below, you can find features that are implemented in the level select package:

- Complete level selection environment.
- Support for score calculation and visualization (stars)
- Automatically tracks level progression, persistent throughout sessions.
- Supports any amount of levels, pagination automatic.

3. Set-Up

Level Select Handler

To start implementing level selection, you first have to create a new class that inherits from LevelSelectHandler. This is an abstract class that requires a few generics and abstract methods to be implemented.

Generics

In total, the LevelSelectHandler requires 3 generics; TConfig, TResult, TMinigame. Below you can find an implementation example and their definitions.

```
public class BubbleShooterLevelSelectHandler : LevelSelectHandler<BubbleShooterConfig, BubbleShooterResult, BubbleShooterManager>
```

Generic	Desciption
TConfig	This is your configuration scriptable object type for your game.
TMinigame	This should be the type of your game implementation. Your game manager class.
TResult	Result object of your game.

Abstract Methods

LevelSelectHandler requires 2 abstract methods to be overridden to fit your game purpose, CalculateScore and GetConfig. Below you can find an implementation example and their definitions

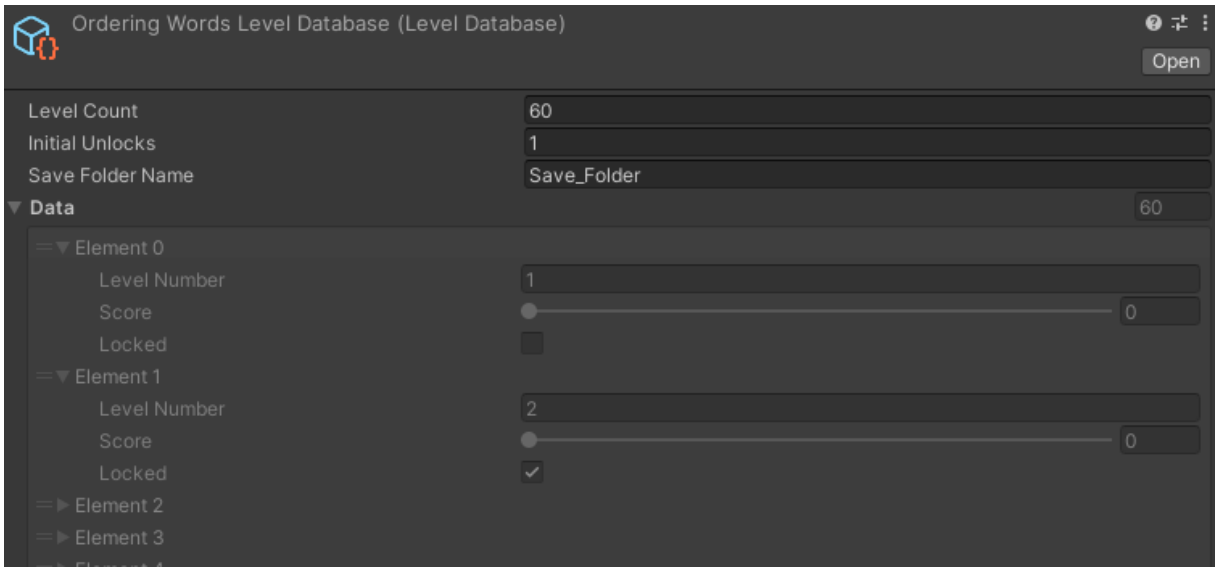
```
/// <summary>
/// <inheritdoc/>
/// </summary>
/// <param name="result"><inheritdoc/></param>
protected override float CalculateScore(BubbleShooterResult result) => Mathf.Clamp( value: result.score, min: 0, max: 1);

/// <summary>
/// <inheritdoc/>
/// </summary>
/// <param name="levelNumber"><inheritdoc/></param>
/// <returns><inheritdoc/></returns>
protected override BubbleShooterConfig GetConfig(int levelNumber) => _levelConfigs[Mathf.Clamp( value: levelNumber, min: 0, max: _levelConfigs.Length - 1)];
```

Method Signature	Description
protected TConfig GetConfig(int levelNumber)	Method implementation should return a config object based on the given level. The level number can be used here to pace level difficulty.
protected float CalculateScore(TResult result)	Method implementation should calculate a score based on the results of the game. The float value returned should be between a range of 0 and 1.

Level Database Scriptable Object

LevelDatabase ScriptableObject. This ScriptableObject will store your level data and make it persistent. This will initially reserve 60 slots for level data. You can add more by increasing the Level Count.



4. API

LevelSelect

The entry point for level selection in games.

Property Name	Type	Description
SCENE_NAME	string	Name of the level select scene
SelectedLevel	LevelButton	The level that is currently selected
LevelSelected	Action<LevelData>	Called when a level is selected

Method name	Return Type	Parameters	Description
Populates	void	LevelDatabase	Populates the level select environment with a given level database.
DisplayPageLevels	void	pageNumber	Display levels for the given page number.
OnLevelSelected	void	LevelButton	Called when a level is selected.

LevelSelectHandler

Can be inherited from to be able to add a level select flow to your game.

Will start the level select UI and allow the user to use the level selection to start a game.

Property Name	Type	Description
LevelSelectOpened	Action	Level select UI was oppened
LevelSelectClosed	Action	Level select UI was closed
CurrentLevel	int	The current level the user is playing

Method name	Return Type	Parameters	Description
OnLevelSelected	void	LevelData	Called when a level has been selected. Start at that level.
OnMinigameFInished	void	TResult	Called when the game has finished. Save the result to the database and unlock the next level.
HideLevelSelect	void	-	Hide the level select UI
ShowLevelSelect	void	-	Show the level select UI
GetCongig	TConfig	int	Return the level config object for that level
CalculateScore	float	TResult	Transform the TResult of the game into a score. Score is a float between 0 and 1.

PageDotUI

Can be inherited from to be able to add a level select flow to your game.

Handles the UI for the selectable dot in the pagination menu.

Property Name	Type	Description
Color	Color	The color of the page dot.
Index	int	The index of this dot
Rect transform	RectTransform	The rect transform of this UI element
OnClick	Action	Called when the dot is clicked

Method name	Return Type	Parameters	Description
SetDotActive	void	bool	Setthe dot to be the active dot.
OnButtonClicked	void	-	Call the OnClick event

PageSelectorUI

Can be inherited from to be able to add a level select flow to your game.

Manages the UI for making a page selection.

Property Name	Type	Description
PageSelected	Action<int>	Called when a page is selected.

Method name	Return Type	Parameters	Description
Populate	void	Pagination,int	Populates the page selection based on the given pagination object
SetPageActive	void	int,boot	Set the given page to be active

Pagination

Can be inherited from to be able to add a level select flow to your game.

Main pagination logic controller.

Property Name	Type	Description
PageChanged	Action<int>	Called when a page is changed, passing the new page number.
ControlsEnabled	Action<bool>	Called when the control enabled state has changed, passing the new state.
CurrentPage	int	Current page index.
TotalPages	int	Total amount of pages.
ItemsPerPage	int	Total amount of items per pages.
LastPage	int	The last page that was used before the current page.
ControlsAreEnabled	bool	Whether the pagination controls are enabled and can moover between pages.

Method name	Return Type	Parameters	Description
SetPage	void	int pageNumber	Set the page to a new page number.
NextPage	void	-	Go to next page.
PreviousPage	void	-	Go to the previous page.
Populate	void	int totalPages, int itemsPerPage	Populate the pagination with the given amount of pages and item per page.

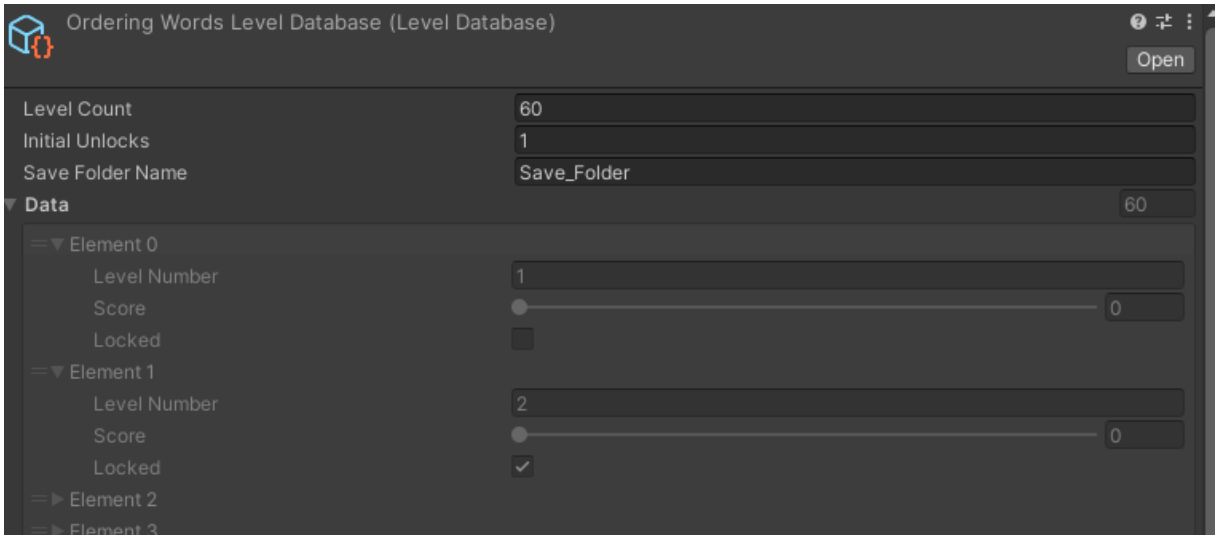
5. How to configure the level database.

In the level select package, to add a level, you have to change 2 components.

- The Level select handler class must return a new GameConfiguration object for the newly created index.
- The size of the Level Database scriptable object.

Level Database configurations

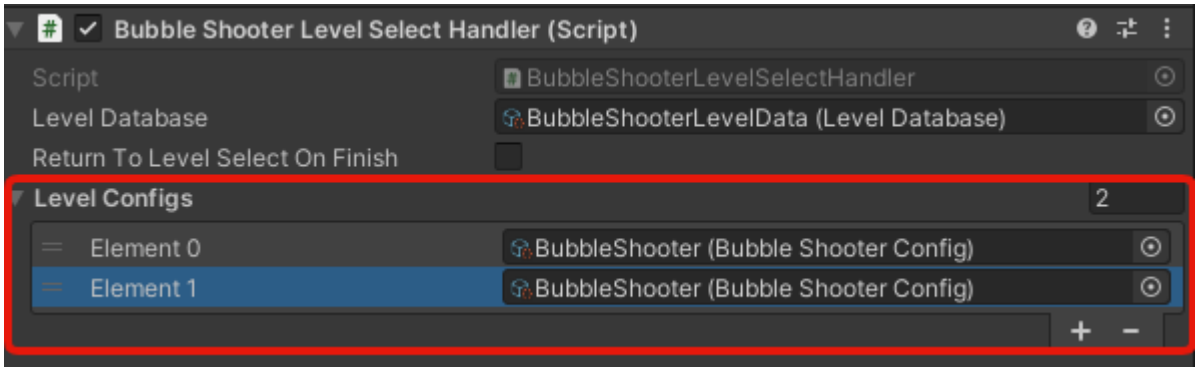
To increase the amount of levels the user can play, 'Level Count' needs to be raised. To increase the initial levels that are unlocked, 'Initial Unlocks can be raised. Keep in mind that this numbers needs larger than one.'



Game configuration object

The second step is to link that button to your game configuration object. For that you can create a new GameConfig. And make sure that the GetConfig(int levelNumber) method of your level select handler implementation is returning the newly made configuration.

To do so, you usually give a list of gameConfig objects to the levelSelectHandler and return GameConfigList[levelNumber] in the get config method. You then just need to manage the list as you see fit. You can see an example of that implementation below.




```
/// <summary>
/// The _levelConfigs field contains the configurable levels for each level for level selection.
/// </summary>
[SerializeField]
private BubbleShooterConfig[] _levelConfigs;  Serializable

/// <summary>
/// <inheritdoc/>
/// </summary>
/// <param name="levelNumber"><inheritdoc/></param>
/// <returns><inheritdoc/></returns>
protected override BubbleShooterConfig GetConfig(int levelNumber) => _levelConfigs[levelNumber];
```

6. How to determine your score

Score calculation needs to be done in your implementation of the LevelSelectHandler in the method CalculateScore.

The method is receiving your game result object and returning a float between 0 and 1. You can set up your own score formula to transform your result into this float. Below is an example of score calculation for a race game.

7. WebGL

Unity does **not** support saving files in WebGL. Therefore, a JavaScript was used to handle saving and loading of the level data on this platform.

8. Support and feedback

If you have any questions regarding the use of this asset, we are happy to help you out. Always feel free to contact us at:

asset-support@d-tt.nl

(We typically respond within 1-2 business days)

We are actively developing this asset, with many future updates and extensions already planned. We are eager to include feedback from our users in future updates, be they 'quality of life' improvements, new features, bug fixes or anything else that can help you improve your experience with this asset. You can reach us at the email above.

Reviews and ratings are very much appreciated as they help us raise awareness and to improve our assets.

DTT stands for Doing Things Together

DTT is an app, web and game development agency based in the centre of Amsterdam. Established in 2010, DTT has over a decade of experience in mobile, game, and web based technology.

Our game department primarily works in Unity where we put significant emphasis on the development of internal packages, allowing us to efficiently reuse code between projects. To support the Unity community, we are publishing a selection of our internal packages on the Asset Store, including this one.

More information about DTT (including our clients, projects and vacancies) can be found here:

https://www.d-tt.nl/en/