ByChance for Unity

The ByChance Framework allows game developers to provide an infinite amount of unique levels for both 2D and 3D games. Easy to integrate into video games of all genres, ByChance enables you to generate complex levels including all of their game components and comes with useful post-processing algorithms that can be applied afterwards in order to ensure a great gaming experience.

Contents

Getting Started............................................................................................................................................. 3
  Project Structure....................................................................................................................................... 3
Generating Your First Level.......................................................................................................................... 4
Customizing the Chunk Library.................................................................................................................. 7
  Chunk Rotations....................................................................................................................................... 7
  Chunk Weights......................................................................................................................................... 7
  Anchors..................................................................................................................................................... 7
Configuring the Level Generator.................................................................................................................. 8
  Restricting Context Alignment.................................................................................................................. 8
  Aligning Adjacent Contexts..................................................................................................................... 9
  Discarding Open Contexts....................................................................................................................... 9
  Filling Contexts........................................................................................................................................ 9
Adapting the Level Generation Process....................................................................................................... 10
  Setting the First Level Chunk................................................................................................................ 10
  Enabling and Disabling Logging............................................................................................................... 10
  Using Level Generator Seeds.................................................................................................................. 10
  Required Anchors.................................................................................................................................... 11
Offline Level Generation............................................................................................................................. 11
Extending ByChance for Unity..................................................................................................................... 12
  Termination Conditions............................................................................................................................ 12
  Chunk Distribution.................................................................................................................................. 12
  Chunk Pre-Conditions............................................................................................................................... 13
  Level Validation....................................................................................................................................... 13
Next Steps................................................................................................................................................... 14
Upgrading From Previous Versions............................................................................................................... 15
  Version 1.0 to 1.1...................................................................................................................................... 15
  Version 1.1 to 1.2...................................................................................................................................... 15
Version 1.2 to 1.3 ........................................................................................................15
Version 1.3 to 1.4 ........................................................................................................15
Version 1.4 to 1.5 ........................................................................................................15
Version History .............................................................................................................16
Version 1.5.1 ................................................................................................................16
  Bug Fixes .....................................................................................................................16
Version 1.5 ....................................................................................................................16
  Features .......................................................................................................................16
Version 1.4 ....................................................................................................................16
  Scripts .........................................................................................................................16
  Features .......................................................................................................................16
  Improvements .............................................................................................................16
  Bug Fixes .....................................................................................................................17
Version 1.3 ....................................................................................................................17
  Features .......................................................................................................................17
  Bug Fixes .....................................................................................................................17
Version 1.2 ....................................................................................................................17
  Features .......................................................................................................................17
  Improvements .............................................................................................................17
  Bug Fixes .....................................................................................................................17
Version 1.1 ....................................................................................................................17
  Features .......................................................................................................................17
  Improvements .............................................................................................................17
  Bug Fixes .....................................................................................................................18
Version 1.0 ....................................................................................................................18
Getting Started

The core of the ByChance Framework is a generic level generation algorithm that is able to construct 2D and 3D levels alike. Thus, you only need to understand the framework fundamentals once, and will then be able to create levels for games of all genres.

ByChance for Unity adds a layer on top of the ByChance Framework itself, allowing you to create and customize your preferred level generator just by attaching scripts and using inspectors. By making extensive use of Unity Editor features like prefabs, custom editors, handles and native Undo/Redo, ByChance for Unity lets you create levels just like you’re already used to – the Unity way.

Project Structure

To get started, just import the ByChance Unity package into your project. Now, the ByChance folder should contain the following directory structure:

- **Example**
  - **ByChance.** Example chunk library and level building blocks.
  - **Prefabs.** Game objects that are spawned throughout the generated levels.
  - **Scripts.** Code that is specific to this showcase.
  - **Sprites.**Sprites that are specific to this showcase.
  - **Example Scene.** Example scene that generates random levels.

- **Framework**
  - **Editor**
    - **Data.** Custom ByChance tag manager.
    - **Editors.** Custom inspectors for ByChance MonoBehaviours.
    - **Images.** Framework-specific images, such as the script editor icons.
    - **PropertyDrawers.** Custom drawers for ByChance properties.
    - **Util.** Editor utility classes that provide common editor behavior, such as Undo & Redo.
    - **Windows.** Code for menu items that have been added to the Window > ByChance Framework menu.
  - **Scripts**
    - **Configuration.** ByChance Framework configuration classes, independent of Unity3D.
    - **Core.** ByChance Framework core implementation, independent of Unity3D.
    - **Levels3D.** ByChance Framework 3D level implementation, independent of Unity3D.
    - **Unity.** Unity3D layer for the ByChance Framework. Add these behaviours to your game objects.
    - **Unity/Extension.** Extensibility interface for ByChance. Extend these classes to define your very own custom level generation behavior, if required.
    - **Unity/Internal.** Interface between ByChance core implementation and the Unity3D layer. You won’t need to understand these classes, nor work with them explicitly.
    - **Util.** Framework utility classes, e.g. for working with collections.
Generating Your First Level

ByChance thinks of a game level as a bounded space consisting of a limited number of level building blocks called *chunks*. Each chunk contains information about its extents, its position and rotation as well as about where to align it to the existing level and where to add game elements like enemies, items or levers.

The only thing you need to do before generating your first level is to setup a *chunk library*. This chunk library holds a set of *chunk templates* that is used by the level generator for generating the game levels.

1. Click **Window > ByChance Framework > Create Chunk Library**.

Congratulations, you’ve created your first, empty chunk library! Now, you need to add chunk templates to be used by the level generator.

   1. Click **GameObject > 3D Object > Plane**.
      2. Rename the Plane game object to **Room**.
      3. With the Room object still selected, click **Window > ByChance Framework > Create Chunk Template** (or press CTRL+SHIFT+T).

Note that your Room prefab now has the ByChance Chunk Template behavior attached. This effectively makes your Room a level building block that can be used by the level generator for creating your levels.

Let’s add a little more information for the level generator. The generator needs to know how chunks constructed with this template can be put together. ByChance uses the concept of *contexts* for specifying where chunks can be aligned. Every chunk has to contain at least one single context describing the relative position at which it may be aligned to other chunks:

   4. In the Chunk Template inspector, just below the **Add Context** button, click **Left Context, Right Context, Front Context** and **Back Context**.
   5. Create a prefab by dragging the Room into your project pane.
Now, our chunk template has four contexts, one at the center of each side. These contexts are indicated by white spheres in the Scene view and can be dragged around as you like.

Note the thin white box around the plane that indicates the bounds the level generator will use for placing your Room. The extents have been automatically computed by the framework when you created the chunk template. You can easily update them at any time by clicking Compute Extents From Renderers.

Almost there. There are just two steps left: We need to add the template to our chunk library, and we need to setup the actual level generator.

1. Select the New ByChanceChunkLibrary object in your project pane.
2. In the Chunk Library inspector, drag your Room prefab to the first template in the list.

While the framework will work with a chunk library that contains only one chunk template, the resulting level is sure to be dull. The more choices the framework has in regard to chunk templates, the more interesting the final result will be.

As soon as the chunk library has been properly set up, it can be passed to the level generator, along with the desired level size:

1. Click Window > ByChance Framework > Create Level Generator.
2. Select the Level Generator game object in your scene, and assign your Chunk Library to your level generator.
3. In the ByChance Level Generator inspector, check Generate On Start.
4. In the inspector, set Level Extents to 30 / 1 / 50.
5. Remove the Room game object from your scene and save the scene.
When you click the Play button in the Unity Editor and switch to the Scene view, you will see that your Rooms have been instantiated by the level generator and parented to the Level Generator game object!

The level still looks a little boring, so let’s exit Play mode and add at least one more chunk template:

1. Click GameObject > 3D Object > Plane.
2. Select the Plane game object and set its Scale X value to 0.5.
3. Rename the Plane game object to Floor.
4. With the Floor object still selected, click Window > ByChance Framework > Create Chunk Template (or press CTRL+SHIFT+T).
5. In the ByChance Chunk Template inspector, just below the Add Context button, click Front Context and Back Context.
6. In the same inspector, check Allow Chunk Rotation and create a prefab by dragging it into your project pane.
7. Select your NewByChanceChunkLibrary.
8. In the Chunk Library inspector, increase the size of the list of templates to 2 and drag your Floor prefab to the second template in the list.
9. Remove the Floor game object from your scene and save the scene.

When you click the Play button now, you will see that your Rooms and Floors have been instantiated and connected by the level generator.

Congratulations! You’ve created your first random level!
Customizing the Chunk Library

Your level might not show all of your desired characteristics yet. For instance, rooms are connected to rooms, and floors can lead nowhere. While some games may tolerate (or even like) this behavior, others might not.

Luckily, there are a lot of parameters that can be tweaked to generate levels exactly the way you want them to be.

Chunk Rotations

Apart from assigning contexts to chunk templates, the ByChance Framework also offers the option to allow the rotation of chunks during the level generation process. This can be very useful when defining floor or corner chunks and can keep the number of chunk definitions at a minimum.

Just check Allow Chunk Rotation in the ByChance Chunk Template inspector to allow the level generator to rotate the chunk during the level generation process.

ByChance always rotates by 90° in order to reduce the number of iterations. 3D chunks are always rotated by 90° around the y-axis. If the chunk has been rotated by 360° around the y-axis, it is rotated by 90° around the x-axis after. If the chunk has been rotated by 360° around the x-axis, it is rotated by 90° around the z-axis after. This leads to a total of 64 possible rotations of each 3D chunk. You might want to consider adding rotated versions of the chunk to the chunk library instead in order to trade memory for running time.

Chunk Weights

Every chunk has a relative weight that tells the level generator how often a specific chunk should be added to the level. A chunk with a weight of 3 is added to the level about three times as often as a chunk with a weight of 1. You can specify these weights by selecting your chunk library and setting the Weight value of each template in the Chunk Library inspector.

Anchors

Every chunk may contain one or more tagged anchors describing the relative position at which game elements can be added. To add an anchor, just select your chunk template and click Add Anchor in the ByChance Chunk Template inspector. You can specify its relative position in the inspector, or use the editor handles in the Scene view instead.

These anchors can be automatically filled by the level generator after the level has been successfully constructed:

1. Click Window > ByChance Framework > Tag Manager and add as many Anchor Tags as you like.
2. Select your Level Generator.
3. Click Add Component > Fill Anchor.
4. Set the Anchor Tag to the tag of the anchor that should be filled by Unity.
5. Click Add Anchor Prefab to specify a new prefab that is allowed to fill the anchor.
6. Set the Probability to a value between 0 and 1 to have the level generator fill the specified anchor with the passed game object with the specified probability.
Configuring the Level Generator

For more advanced scenarios, the level generator can be configured with additional parameters. The following sections summarize the possibilities of customizing the level generation process.

Restricting Context Alignment

If your chunk library contains many chunks with very similar extents, like the rooms in our very first level, this can lead to boring repetitive patterns.

In the Angry Bots Infinite showcase (before ByChance 1.2), we decided to tag door contexts in order to avoid artifacts of this kind: Doors that were intended to lead to other rooms were prevented from being aligned to each other.

1. Click Window > ByChance Framework > Tag Manager and add as many Context Tags as you like.
2. Select your Level Generator.
3. Click Add Component > Restrict Context Alignment.
4. Set the First Context and Second Context fields to the tags of the contexts you don’t want to be aligned to each other.

Hint: You can add any or all of the configuration scripts mentioned in this section to the level generator game object, or to any game object below the level generator in the scene hierarchy.
Aligning Adjacent Contexts
You can have the level generator check all open contexts in the level and connects pairs of contexts that are within a certain offset to each other. The probability of this to occur increases if your chunks have similar dimensions and their contexts have similar relative positions.

1. Select your Level Generator.
2. Click Add Component > Align Adjacent Contexts.
3. Set the Maximum Offset field to the maximum distance between two contexts to be automatically aligned by the level generator, in Unity units.

Discarding Open Contexts
Finally, you can let all open contexts be cleaned up. This is useful for discarding unused contexts before having them filled.

1. Select your Level Generator.
2. Click Add Component > Discard Open Contexts.
3. Set the Chunk Template field to the template of the chunks to discard all open contexts of.

Filling Contexts
Just like anchors, contexts can be automatically filled by the level generator after the level has been successfully constructed:

1. Click Window > ByChance Framework > Tag Manager and add as many Context Tags as you like.
2. Select your Level Generator.
3. Click Add Component > Fill Context.
4. Set the Aligned Context Prefab to the prefab to instantiate at the position of the context if it has been successfully aligned to another one. This could be a Door prefab, for example.
5. Set the Discarded Context Prefab to the prefab to instantiate at the position of the context if it has been discarded by a post-processing step. This could be a Wall prefab, for instance.
6. Set the Context Tag field to the tag of the contexts to fill with the specified prefabs.
Adapting the Level Generation Process

Setting the First Level Chunk

In some cases, you don’t want the framework to pick the first level chunk for you. Let’s assume that you want the level generation process to start with the chunk containing the spawn point of the player character. Then you can set this chunk as initial level chunk as follows:

1. Select your Level Generator.
2. Set the Initial Chunk field to the prefab of the chunk template the generator should start with.

Enabling and Disabling Logging

If you want to log the details of the level generation process, you can enable logging at the ByChance Level Generator inspector. Note that logging slows down the level generation by a big deal, which is why we recommend disabling logging in production:

1. Select your Level Generator.
2. Check or uncheck Log Level Generation.

Using Level Generator Seeds

If you want to generate a given level again, you can make use of the level seed. This seed is written to the Unity console if logging is enabled, and it can be passed to the level generator at the ByChance Level Generator inspector:

1. Select your Level Generator.
2. Set the Seed field to the seed of the level to generate.
Required Anchors
Some games have minimum requirements for each level, such as the spawn position for the player and the level exit. You can have ByChance throw away a level and start over if these anchors are not present as follows:

1. Select your Level Generator.
2. Click Add Component > Required Anchor.
3. Set the Required Anchors field to the tags of the anchors that are mandatory for valid levels.

Note that this can lead to the halting problem: If the specified anchors aren’t present in your Initial Chunk, technically ByChance could produce levels indefinitely before finding a valid one. Reduce the amount of required chunks to a minimum.

If you want to ensure chunks of specific type are present, or a certain amount of total level chunks, use the Required Chunk or Required Chunk Count scripts, respectively.

Offline Level Generation
One last thing: You even can generate levels in editor mode, without having to start the game:

1. Select your Level Generator.
2. Open the script context menu.
3. Click Generate Level.

The level generator will create the level chunk objects in the scene and fill their anchors. This allows level designers to pump out a lot of random levels, while being able to tweak them to their needs.
Extending ByChance for Unity

There are several ways of cleanly extending the existing functionality of the framework in order to generate levels exactly the way you want them to be. This section summarizes the different ways of customizing ByChance for Unity.

All extension mechanisms are found in the Framework/Scripts/Unity/Extension folder. Deriving from any script of that folder and adding it to the level generator game object (or any game object below in the scene hierarchy) will automatically add its behavior to the level generation process.

Termination Conditions

By default, the level generation stops as soon as the level can’t be expanded any more in any direction, or more precisely, as soon as no chunk template from the chunk library can be added to any context.

However, you might need a different (earlier) condition to stop level generation, for instance after having added a specific amount of rooms to your dungeon:

2. Implement the method ConditionIsMet.
3. Add the new script your level generation game object (or any game object below in the scene hierarchy).

You will need to cast the passed level to Level3D in order to access any of its properties.

The level generation will stop as soon as any termination condition is fulfilled.

Take a look at the ByChanceMaximumChunkCount script to get a head start.

Chunk Distribution

To grow your level, ByChance will filter your chunk library for candidates for a free context in the level, and pick a random chunk according to its weight in the library.

Sometimes though, you might want to have more sophisticated rules for expanding the level. One example might be to add rooms with increasing difficulty, scaling by distance from the level start:

2. Implement the method GetEffectiveWeight.
3. Add the new script your level generation game object (or any game object below in the scene hierarchy).

ByChance will pass the context to expand the level at first, the context of the candidate to add to the level there second, and the whole current level third. Note that you will need to cast the passed level to Level3D again in order to access any of its properties.

With this data, you can customize your level chunk distribution as much as you like. Note that adding multiple of these scripts to your level generator will have ByChance ignore all but one of them.

You may take a look at the implementation of the base class to get started.
Chunk Pre-Conditions
Some puzzles in your game might require multiple anchors. Obviously, it might be much more interesting if these anchors are part of different chunks, at different positions in your level. Think of keys matching specific locked doors or chests, for example.

With ByChance, you can prevent a specific chunk to be added to your level unless an arbitrary condition holds:

2. Implement the method `CanBeAligned`.
3. Add the new script your level generation game object (or any game object below in the scene hierarchy).

The level generator will pass the two contexts to be aligned, as well as the current generated level. You can then check the level for any arbitrary condition, and reject the context alignment if necessary, effectively ruling out the chunk from being added to the context at this time. Note that you will need to cast the passed level to `Level3D` again in order to access any of its properties.

For two contexts to be aligned, all of these conditions must hold.

Taking a look at the `ByChanceRestrictedChunk` script should get you started.

Level Validation
You have already heard of required anchors or chunks before. Adding these to the level generator will reject a generated level unless a specific validation rule is passed, and start over.

Apart from the validation rules shipped with ByChance, you can define your own rules as you like:

2. Implement the method `ValidateLevel`.
3. Add the new script your level generation game object (or any game object below in the scene hierarchy).

After a level is generated, ByChance will pass it to all validation rules it can find. The level will be discarded if any of these rules is violated.

The `ByChanceRequiredChunkCount` script might be a good place to start.
Next Steps

You’ve learned how to integrate the ByChance Framework into your game and how to have the level generator create random levels the way you want them to be.

We recommend taking a look the ByChance/Example/ Example Scene.unity and the chunk library and chunks in the ByChance/Example/ByChance folder to get an idea of what’s possible with the framework.

In case you run into any issues, head over to our issue tracker and we’ll investigate immediately: https://github.com/npruehs/ByChance/issues. If you need help, don’t hesitate to ask a question at http://www.levelsbychance.com/help!

Finally, when you’re finished creating your awesome game with ByChance, we’d love to hear from you: dev@npruehs.de!
Upgrading From Previous Versions

Version 1.0 to 1.1
Since ByChance for Unity 1.1, chunk libraries are stored as scriptable objects instead of prefabs. Furthermore, a new tag manager helps assigning correct anchor and context tags.

In order to upgrade your project from version 1.0 to 1.1, just follow these steps:

1. Create a backup of your project.
2. Click Window > ByChance Framework > Create Chunk Library.
3. Copy all entries from your existing chunk library to the new one.
4. Delete your old chunk library.
5. With your new chunk library selected, click Window > ByChance Framework > Upgrade > 1.0 -> 1.1.
6. Save your project to apply the changes made to the tag manager asset.
7. Select your level generator.
8. Verify all anchor and context tags of your scripts are correctly assigned. If this is not the case, chances are that you’ve selected it before, losing all tags due to the new framework structure. Just revert the scene containing your level generator, and select it again. Everything should be fine now.
9. Assign your new chunk library to your level generator.
10. Re-assign your chunk templates to any Discard Open Contexts scripts you’re using.

If you’re experiencing any issues upgrading your project, head over to http://www.levelsbychance.com/help at any time. We’re happy to help!

Version 1.1 to 1.2
Version 1.2 contains no breaking changes. You’re good to go!

Version 1.2 to 1.3
Version 1.3 contains no breaking changes. You’re good to go!

Version 1.3 to 1.4
Version 1.4 contains no breaking changes. You’re good to go!

Version 1.4 to 1.5
Version 1.5 contains no breaking changes. You’re good to go!
Version History

Version 1.5.1

Bug Fixes
- Generating levels in editor is working again as expected, and shows a progress bar now.
- Seed and level generation info are correctly logged to the console again.

Version 1.5

Features
Added level generation events. The level generator now raises Started, ProgressChanged, Finished and Failed events that allow you to show an appropriate UI, including a progress bar, while the level is being generated.

Version 1.4

Scripts
- Maximum Chunk Count. Finishes level generation after a certain amount of chunks has been added.
- Required Chunk. Chunks that are required for valid levels.
- Required Chunk Count. Minimum total chunk number required for valid levels.
- Restricted Chunk. Prevents a chunk from being placed until a specific anchor has been added to the level.

Features
- Custom termination conditions allow you to stop the level generation before all level space is filled.
- Custom chunk distribution allows you to override the default random chunk selection, for example for adding rooms with increasing difficulty to the level, scaling by distance from the level start.
- Custom chunk pre-conditions prevent a specific chunk to be added to the level unless an arbitrary condition holds, for example for adding locked doors only after the matching key has been added.
- Custom validation rules reject a generated level and have the generator start over, requiring a level to have a minimum number of chunks, for instance.

Improvements
- Configuration scripts don’t need to be put on the level generator game object any more, but may be put anywhere below it.
- Increased performance in the case a level is rejected and a new one is generated with the same configuration.
- Chunk template anchors are added at random initial positions within the chunks instead of the center so they’re more easily moved around in the scene view.
Bug Fixes
- Correctly checking whether 3D chunks fit the depth of 3D level bounds.

Version 1.3
Features
Added the ability to generate levels in editor mode, without having to start game, from the Level Generator context menu.

Bug Fixes
- Correctly rotating chunks around the X-axis.

Version 1.2
Features
Added full support for automatically generating 2D levels! Turn sprite renderers into chunks, and benefit from additional editor support such as automatically rotated contexts while in 2D editor mode.

Improvements
- Providing single buttons for adding centered contexts.
- Automatically computed bounds are never zero in any dimension to prevent infinite level generation.
- Reduced context and anchor sphere handle size in scene view. Moved labels up a bit in order to increase readability.

Bug Fixes
- Correctly re-setting chunk context indices whenever a context is removed.

Version 1.1
Features
- Chunk libraries are stored as scriptable objects instead of prefabs now. This improves the overall memory footprint of the framework.
- New tag manager provides easy access to all anchor and context tags. Scripts that are related to tags in any way now provide dropdown menus instead of textboxes for selecting tags. Users can specify the color illustrating contexts and anchors in the scene view.

Improvements
- ByChance scripts are accessible from the Add Component menu.
- Assigning reasonable default values for context alignment, chunk template extents and level generator properties.
- Removed chunk template creation window, and created simple shortcut for turning a game object into a chunk template.
Bug Fixes
- Showing correct chunk extents in the scene view, even if the chunk is scaled in any dimension.
- Correctly computing the extents of an object, even if it is rotated in any direction.

Version 1.0
Initial release.