JIGSAW PUZZLE KIT

basic manual

Version 1.1

Overview

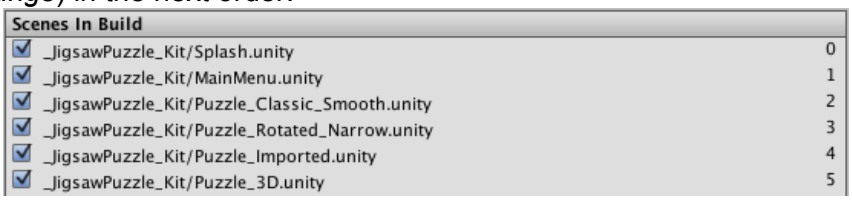
It is a complete solution to create a nice and professional Jigsaw Puzzle game really fast and easy. You can create any puzzle variations /shapes and even in 3D!
The package has everything you may need – flexible puzzle generation and import system, saves system, menus, hints, dialogs, etc. Every element was carefully designed to be effective, universal and easy to use.
Game is fully optimized and Mobile-ready.

The Kit comes with a prepared demo-game that can be easily customized and big amount of useful components/features to create your own gameplay:

- Convenient puzzle generation/import tools
- Gameplay logic and all related components
- Camera controller with Pan&Zoom functionality
- Nice 2D art (including Blue Cartoon GUI, etc)
- Easy and nice dialogs sub-system
- Includes powerful TextureUtility system

How to setup Demo-game

Basic preparations are really simple – just add all scenes to Scenes In Build menu (Main *Menu → File → Build Settings*) in the next order:



Thats all – you can compile game for any platform or just try it in the Editor.

Basic components and their adjustments

Project has next basic objects/components needed for the game (check *_ SystemScripts* folder):

- **PuzzleController** – Prepare and control whole puzzle. Processes pieces movement
- **GameController** – Processing game rules; Handles user interface; Controls all sounds
- **CameraController** – Controls camera Pan and Zoom functionality
- **SimpleDialog** – Processes simple dialogue functionality

Most of basic game objects has prefabs (please check *_Prefabs* folder) – this should help you a lot, since you can just drag'n'drop them to your scene. You may want adjust their behavior or appearance – for this you should either update related values or change Model/Material/etc in the scene.
If you want to get/change any of game asset – you can find their sources in the *DemoAssets* folder.

If you have any questions – please don't hesitate to contact me (e-mail: AllebiGames@gmail.com)
I'm always happy to help you!

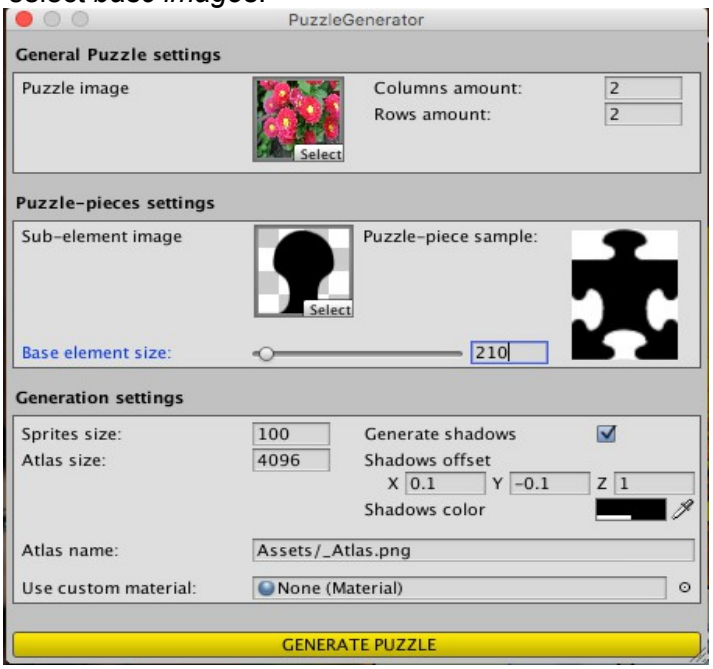
How to create/import new puzzle

System provide an ability to create puzzles in 3 different ways:

1. Full generation
2. Import from layered image (like .PSD)
3. Manual assembling in Unity

Full generation

Kit has built-in script (*_SystemScripts/Editor/PuzzleGenerator_Window.js*) to generate puzzle from flat image and sub-element mask. You can run it from Unity menu (*Main Menu → Tools → PuzzleGenerator*) and select *basc images*:



In the utility window you also can specify:

General puzzle settings

- *Puzzle image* **–** Will be used as main puzzle image
- *Columns amount* **–** Columns number in generated puzzle grid
- *Rows amount* **–** Rows number in generated puzzle grid
-

Puzzle-pieces settings

- *Sub-element image* **–** Contains mask for bumps and holes generation. Image should have minimal size to fit the object(should be X-centered) and use 100% transparency for empty parts.

- *Base-element size* **–** Size of puzzle-piece base (without bumps) Should be bigger than *Sub-element image* size.

Generation settings

- *Sprites size* **–** Max size of generated atlas
- *Atlas size* **–** Sprites resolution (affects actual size of created sprites)
- *Atlas name* **–** Name of creates file (can override existing)
- *Generate shadows* **–** Create fake shadows for pieces or not + shadows settings:
  - Shadows offset – shadow object position offset from puzzle-piece origin
  - Shadows color – shadow color tint (Alpha controls transparency)
- *Custom material* **–** Material that will be assigned to all puzzle-pieces

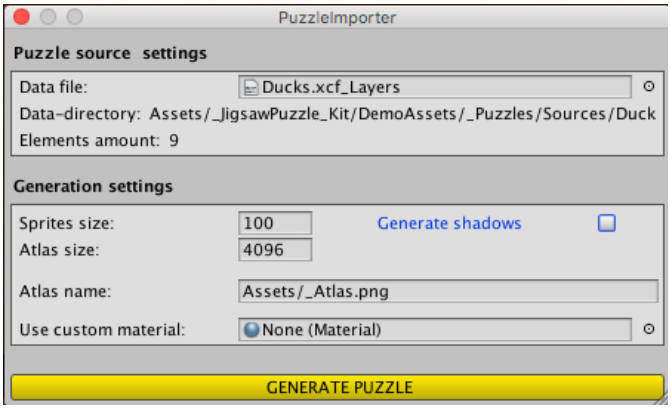
Import from layered image

If you want to import new scene - the easiest way is to create image(.PSD as instance) with layers (each piece should be in separate layer) and then export it to bunch of images + data-file.

Drag those files to Unity Project window and set next ImportSettings for all pieces:

- Texture Type → *Advanced*
- Non Power of 2 → *None*
- Read/Write Enabled → *true*
- Generate Mip Maps → *false*
- Format → *ARGB 32 bit*
- *Max Size* → *4096*

Kit has built-in script (*_SystemScripts/Editor/PuzzleImporter_Window.js*) to create atlas and generate puzzle using this data.
You can run it from Unity menu (*Main Menu → Tools → PuzzleImporter*) and select text *Data file*:



In the utility window you also can specify additional *Generation settings* – they are equal to *Generation settings* described in Full Generation section above.
*Please try to keep **Atlas size** as big as possible (system will automatically reduce it if needed).*

I've prepared a small GIMP script for you (please check *_GIMP _ExtractAllLayers.chm* in _SystemScripts folder) - to export layered image from free "The Gimp" (http://www.gimp.org/) editor really fast and easy. This script will create folder with each layer exported as .png and data file to parse. Drag this folder to your Project window in Unity.

The Gimp can open plenty of file-formats (including .PSD), so if needed - you can save your image to .PSD from Photoshop and then open in GIMP to use exporting script.

More details about using The Gimp for puzzles you can find in section **Create puzzle in GIMP.**

Manual assembling

The most obvious option that allows to create really custom and unique puzzles, but requires a lot of manual work:

- Just create new scene using standard Unity tools and objects. There is only 1 strict requirement: all “puzzle pieces” should have common parent.
- Then add *PuzzleController* component to this parent object.

How to setup scene functionality

Using Prefabs and puzzle importer/generator you can setup everything really easy, though fully manual setup is a bit more complicated. Anyway – follow next simple general steps:

- Create the puzzle using any approach, but all “puzzle pieces” should have common parent
- Don't forget to add **PuzzleController** component to this parent object.
- Add ***GameController*** component any object that will be always active
- Assign to it *Puzzle* property your puzzle object (with **PuzzleController** component)
- Assign assets MUSIC and SOUND SETTINGS of ***GameController***
- *C*reate needed UI elements(or drag and drop **_Canvas** prefab) and add **EventSystem** object from *Main Menu → UI → Event System.*
- Assign UI elements to GUI SETTINGS of ***GameController*** and setup their *OnClick()* events
- Attach **CameraController** component to existing or new main Camera object (or just drag and drop **_MainCamera** prefab to the scene and delete old camera)
- If needed: bring **_DialogWindow** to scene and attach/setup *SimpleDialog* component for camera
- Adjust properties of attached components as you want

Create puzzle in GIMP

1. Add export script to Gimp:

- Copy or move *_GIMP_ExtractAllLayers* script *(stored in proects _SystemScripts folder)* to your GIMP scripts directory. It can be found in the GIMP Preferences: Folders→Scripts.
- Start Gimp
- Make a refresh by using *Filters→Script-Fu→Refresh Scripts* from the image menubar.

The script will now appear in one of your menus (*Filters→User's scripts → Extract all layers)* . If you can't find it, look for it under the root filters menu.

2. Prepare your images for puzzle-pieces

- Obviously, to create scene you should have all your puzzle-pieces images separated(or created separately) for example they can be in different files or as layers in one.

Anyway you'll need to bring all pieces to new .PSD or Gimp file as layers and place them to intended places on scene (background)

- Ensure that each object is in separate layers
- This layer should have minimal size to fit the object:

3. Export scene from Gimp

Run the GIMP script (*Filters→User's scripts → Extract all layers*). The script will create folder with each layer exported as .png and data file to parse.

Important: This will delete all layers from current Gimp file, to bring them back just use *Undo* .

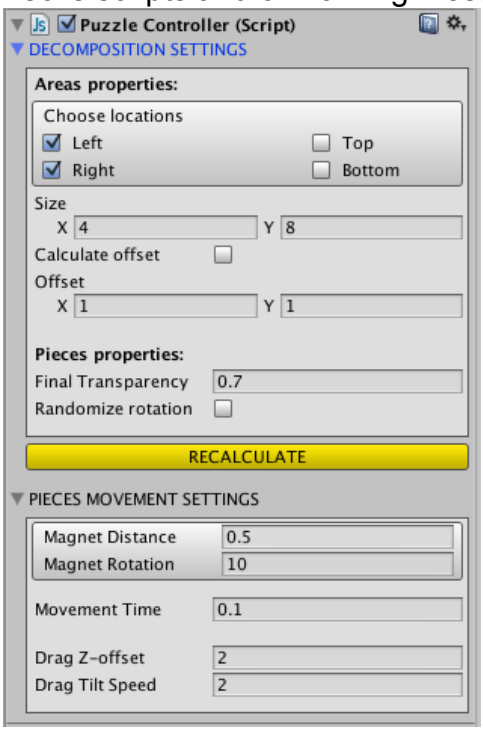
Format of generated data-file (just in case):
- Each layer/piece should be saved to separated *.png* file
- Info about every layer should be saved to text file with structure:

```
Layer_3.png   - layer/file name
723           - X position of layer in the image
790           - Y position of layer in the image
75            - X size of layer (width)
91            - Y size of layer (height)
---------     - separator-string before information about next layer
```

Main components

Whole source code is fully commented to help understand how it works and which functions you may need to use. Below you can find description of main components interface variables (also available as popup hints in Editor).

PuzzleController- Main script to prepare and control whole puzzle. Also processes decomposition and user input (like pieces movement).This component is required for some other components - please checks scripts and/or warning messages.



Left, **Right**, **Top**, **Bottom** - Sides (around puzzle) where pieces will be moved during decomposition
Size - Decomposition areas(where pieces will be located) size
Calculate Offset - Calculate Decomposition areas offset (from puzzle origin)
Offset - Specify manual Decomposition areas offset (from puzzle origin)
FinalTransparency - Change transparency for assembled pieces to this value
RandomizeRotation - Should pieces be rotated during decomposition

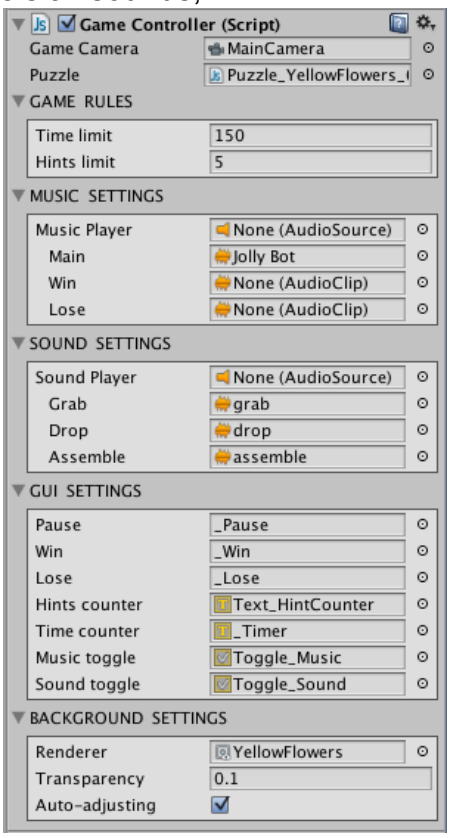
RECALCULATE - Recalculates whole puzzle and prepares it to be used

Magnet Distance - Allowed position offset to consider piece placed to it origin"
Magnet Rotation - Allowed rotation offset to consider piece placed to it origin

Movement Time - Piece needs this amount of time to reach destination during movement
Rotation Speed - How fast piece can be rotated by player

Drag Y-offset - Piece offset(in % of piece size) during dragging by player
Drag Z-offset - Piece offset during dragging by player
Drag Tilt Speed - Piece tilt-speed during dragging by player

GameController - Script contains and processes Win/Lose conditions for the game. Handles user interface; Controls all sounds;



Game Camera - Link to Camera to be used for puzzle (leave empty to try to use Camera.main)
Puzzle - Link to PuzzleController to be processed (leave empty to use attached)

GAME RULES - Contains settings game-rules settings
Time limit - Set time limit for level (-1 to disable)
Hints limit - Set hints limit for level (-1 ro disable)

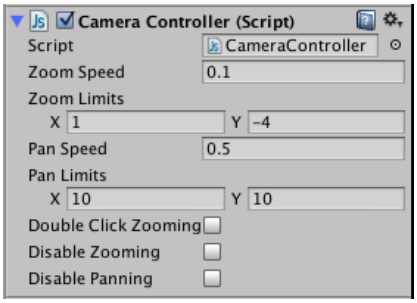
MUSIC SETTINGS - Contains settings related to game music
Music Player - Link to AudioSource component to be used for music (leave empty to create new)
Main - Sound clip to be used as gameplay music
Win - Sound clip to be used as music if player won
Lose - Sound clip to be used as music if player lost

SOUND SETTINGS - Contains settings related to game sound effects
Sound Player - Link to AudioSource to be used for sound effects (leave empty to create new)
Grab - Sound clip will be played when player grabs puzzle-piece
Drop - Sound clip will be played when player drops puzzle-piece
Assemble - Sound clip will be played when player assemble puzzle-piece to puzzle

GUI SETTINGS - Contains settings related to game interface
Pause - Link to object to be shown when game paused
Win - Link to object to be shown when player won the game
Lose - Link to object to be shown when player lost game (if timer enabled)
Hints counter - Link to UI text to visualize remaining hints amount (if hints enabled)
Time counter - Link to UI text to visualize remaining time (if timer enabled)
Music toggle - Link to UI toggle to handle/visualize music enabling/disabling
Sound toggle - Link to UI toggle to handle/visualize sound enabling/disabling
Renderer - Link to background(preview of assembled puzzle) object renderer

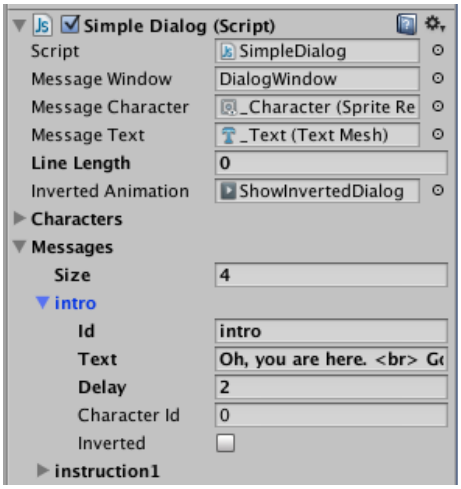
BACKGROUND SETTINGS - Contains settings related to Background (preview of assembled puzzle)
Transparency - Set background transparency
Auto-adjusting - Try to automatically adjust background transform to the puzzle

CameraController - Script controls pan and zoom(pinch) camera to have access to small details in scene



zoomSpeed - Zoom changing speed
zoomLimits - Camera orthographicSize changing limits
panSpeed - Panning speed
panLimits - Camera x,y position changing limits
doubleClickZooming - Enable/Disable Zooming by double-click/tap
disableZooming - Disable Zooming functionality
disablePanning -Disable Panning functionality

SimpleDialog - Script processes simple dialogs functionality



messageWindow - Link to Dialog visualization object
messageCharacter - Link to sprite dedicated for character rendering
messageText - Link to message text mesh
lineLength - Number of symbols in line (before World-wrap)
invertedAnimation - Link to animation to show/make inverted visualization object position
characters - List of characters sprites
messages - List of all messages