Introduction

Texture packing is a process of joining textures together into a single, larger texture called texture atlas. Texture atlases are used in 2D game development to improve rendering performance by reducing the number of draw calls. The process of creating a texture atlas is illustrated in the following picture:

![Texture Atlas Creation](image)

The PaunPacker should meet the following goals:

- Free to use
- Extensible (allowing to create and load plugins)
- Offer additional image processing features (padding, trimming, etc.)
- Have GUI
- Provide a basic toolset for plugin development
- Implement several heuristic algorithms:
  - Minimal rectangles algorithm
  - Genetic based algorithms
  - Bottom-Left algorithm
- Implement alias creation

Solution Approach and Architecture

In the PaunPacker we have decided to use one of the standard approaches to rectangle packing which is to decompose the process of finding a "packing" of the input rectangles into three steps:

1) Minimum bounding box - Finds a minimum bounding box which contains the input rectangles r1, ..., rn. Generally, some sequence of bounding boxes b1, ..., bk could be tested.
2) Place rectangles - Placement of the input rectangles r1, ..., rn into a fixed size bounding box b.
3) Sort rectangles - Sorting of the rectangles according to some criterion.

The main feature of PaunPacker is the ability to parameterize the whole algorithm by the individual steps that could be loaded from a plugin. Therefore instead of only dealing with self-contained algorithms, the algorithms are allowed to consist of three independent parts that are dealt with separately.

The main parts of the software solution are:

1) **PaunPacker.Core library (.NET Standard 2)**
   - Contains implementations of packing algorithms together with some packing related types.
   - Meant to serve as a basic toolset for future plugin development.
   - Allows to reuse the algorithms for general rectangle packing, i.e. not dependent on GUI of the application.
2) **PaunPacker.GUI application (.NET Core 3)**
   - The main application with which the user interacts.
   - Loads and renders the plugins.
   - Outputs texture atlas and metadata obtained from controlling the algorithms loaded from plugins and core library.
3) **Plugins (.NET Standard 2/.NET Core 3)**
   - Contains implementations of extensible components or exports components that are implemented in the core library.

GUI Application

The GUI of the PaunPacker allows the user to adjust various settings, generate texture atlas, process individual images of the texture atlas, load images, manipulate loaded images, etc. and it is shown in the following picture:

![PaunPacker GUI](image)

Not only the packing algorithms are extensible, but also the GUI of the application is extensible because the types exported from plugins could have a view associated with them. Such a view is then rendered into a proper region inside the main window. To simplify working with regions and views they are showing, Prism Framework is used. Prism is also a second option (besides previously mentioned MEF) for loading of plugins but unlike MEF it also performs their initialization, loading and then showing their views inside a corresponding region.

Benchmarks

In addition to the main application for texture atlas generation, a simple CLI application that allows running benchmarks of minimum bounding box finders and placement algorithms has been developed. The benchmarks measure time and average area of the packing result. The benchmarks are generated at runtime for all the types that were loaded from plugins and could be easily instantiated (via IoC resolution). These benchmarks are later compiled using Roslyn and passed to `BenchmarkDotNet` for an execution.

Currently, there are two simple benchmark scenarios:

1) Squares of dimensions 31x1, .... NxN
2) N rectangles of random dimensions

The benchmarks are only expected to be used for a comparison of the individual implementations and not for general comparison of packing algorithms.

Conclusion

The application that has been developed meets all the goals and it is also easily extensible by plugins containing new (parts of) algorithms for packing, tools for image processing and metadata exporters. Also, a simple application that allows performing benchmarks of the individual packing algorithms in order to compare them has been developed. Suggestions for future work include the implementation of new features for enhancing the user experience and optimizing the performance of the implemented packing algorithms.