Coding workshop: install JUCE, create an app and build it in your IDE.

Introduction

This worksheet shows you how to install the JUCE library and the Projucer tool on your system. Then it takes you through building your first application, then customising it.

Objectives

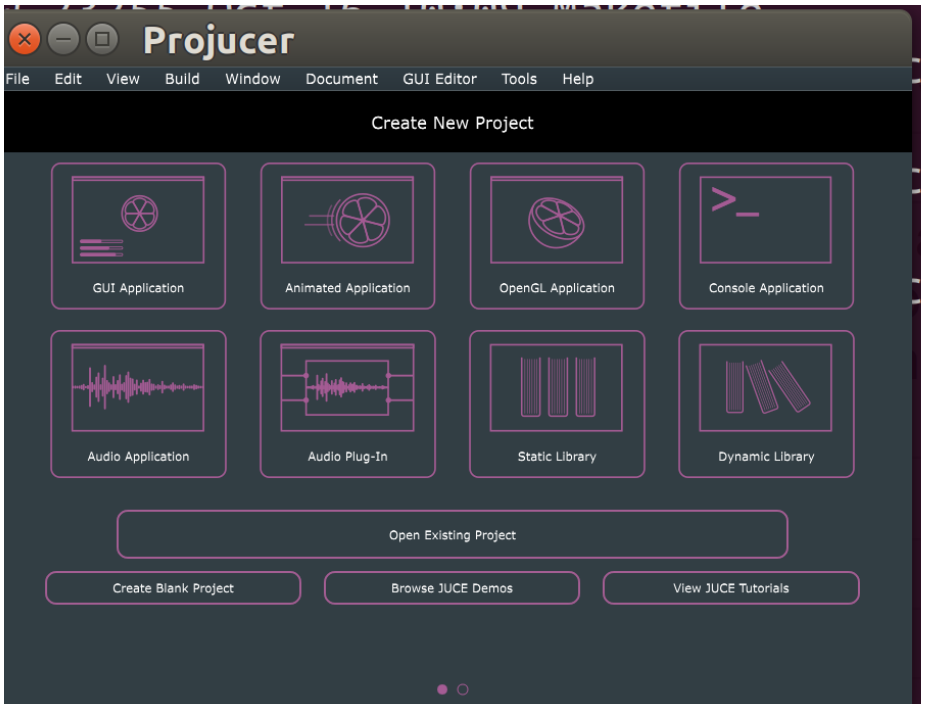
At the end of this worksheet, you should be able to:

- Use the JUCE framework to create and compile a starter application
- Add simple GUI components to a JUCE application

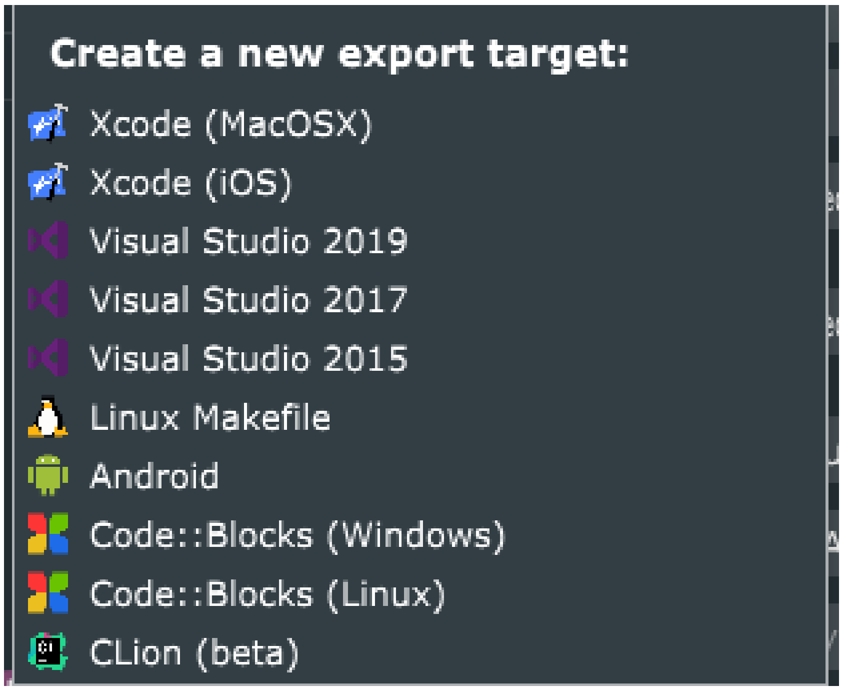
What is Projucer?

Projucer is a tool provided by the creators of JUCE, which makes it easy to create JUCE projects for various IDEs on various platforms.

In the screenshot below, you can see the new project screen on Projucer, showing the different types of project you can create:



In this screenshot, you can see the list of supported IDEs and platforms:



Create a new export target:

- Xcode (MacOSX)
- Xcode (iOS)
- Visual Studio 2019
- Visual Studio 2017
- Visual Studio 2015
- Linux Makefile
- Android
- Code::Blocks (Windows)
- Code::Blocks (Linux)
- CLion (beta)

Installing Projucer

Projucer works on Windows, Mac and Linux.

Before installing Projucer, you will need an IDE installed.

- Windows users: the recommended IDE is Visual Studio 2019.
- Mac users: the recommended IDE is Xcode.
- Linux: you can use any IDE that supports Makefiles. We use Visual Studio Code on Linux.

Once you have an IDE installed, go to the JUCE website: https://juce.com/

and download the installer. Run the installer, then run the Projucer application.

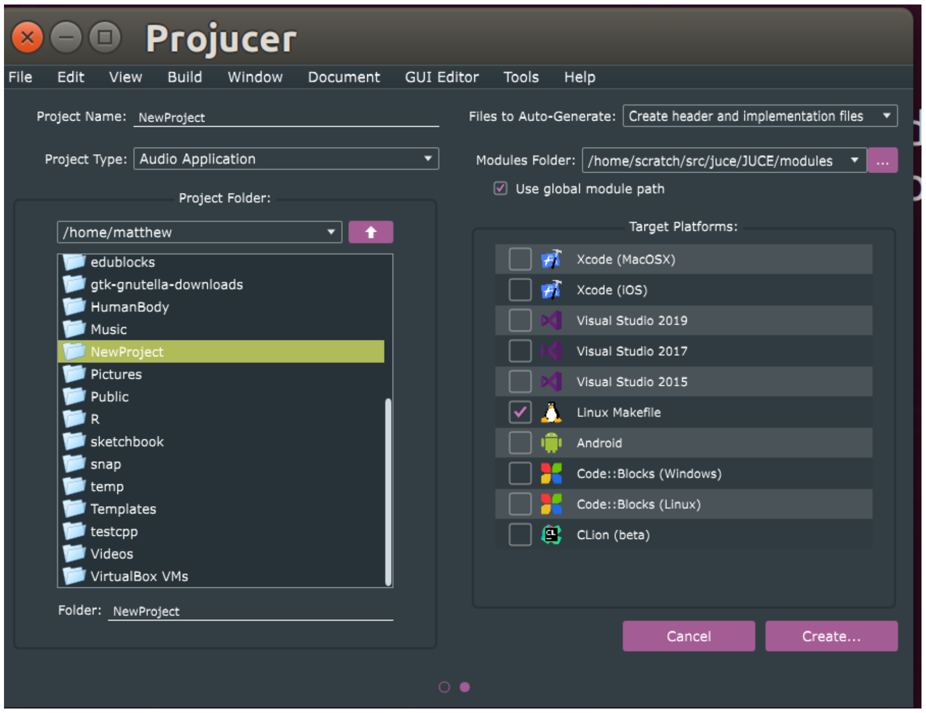
You will need to register for a free JUCE account when you run the Projucer application.

Create your first JUCE project

Once you have an IDE and Projucer installed and you have registered for a JUCE account, you are ready to create your first JUCE project.

Inside Projucer, from the File menu, select New Project. Select Audio application from the available options.

Once you have selected the project type, you should see the New Project screen, like this:



Project Name: NewProject

Files to Auto-Generate: Create header and implementation files

Project Type: Audio Application

Modules Folder: /home/scratch/src/juce/JUCE/modules

[✓] Use global module path

Project Folder:

/home/matthew

- edublocks
- gtk-gnutella-downloads
- HumanBody
- Music
- NewProject
- Pictures
- Public
- R
- sketchbook
- snap
- temp
- Templates
- testcpp
- Videos
- VirtualBox VMs

Folder: NewProject

Target Platforms:

- [ ] Xcode (MacOSX)
- [ ] Xcode (iOS)
- [ ] Visual Studio 2019
- [ ] Visual Studio 2017
- [ ] Visual Studio 2015
- [✓] Linux Makefile
- [ ] Android
- [ ] Code::Blocks (Windows)
- [ ] Code::Blocks (Linux)
- [ ] CLion (beta)

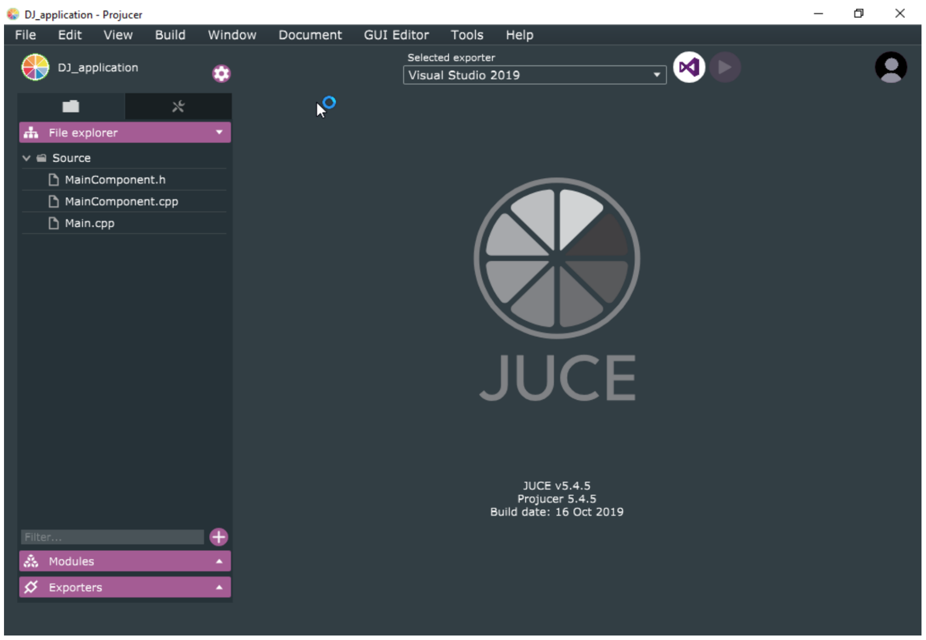
- Give the project a name in the ‘Project Name’ field, for example, ‘DJ_application’.
- Select the folder where you want to create the project.
- Select the exporters you want from the list on the right.
- When all the settings are how you want them, click the create button.

Building the project

Once you have created the JUCE project, you can open it in your IDE and build it. Here are platform-specific instructions for doing that.

Windows

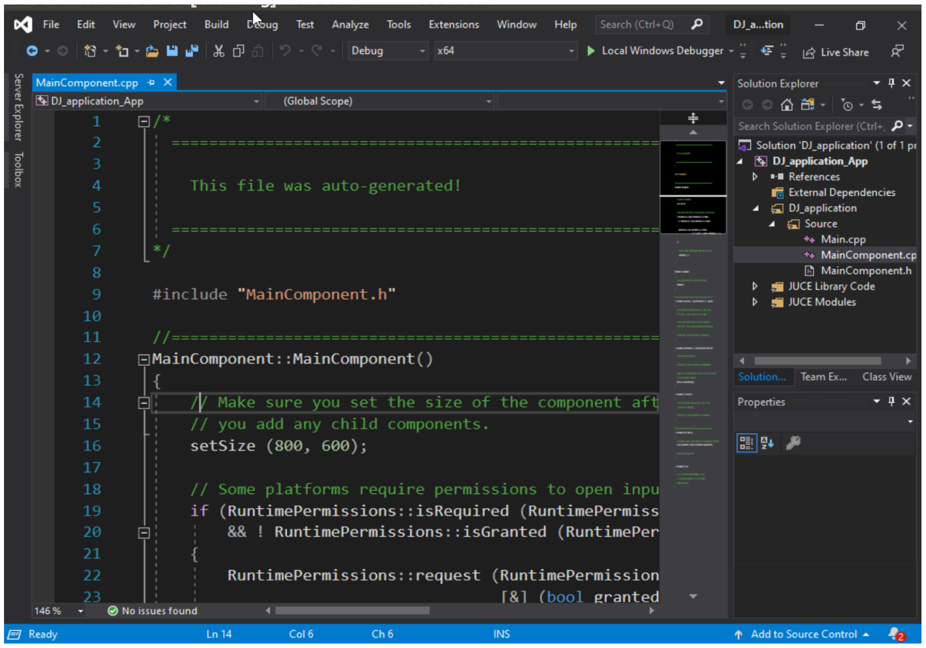
In Projucer, with your new project visible, as follows:



- Click the selected exporter menu at the top of the Projucer window and select Visual Studio.
- Click the Visual Studio icon next to the drop-down menu to open the project in Visual Studio

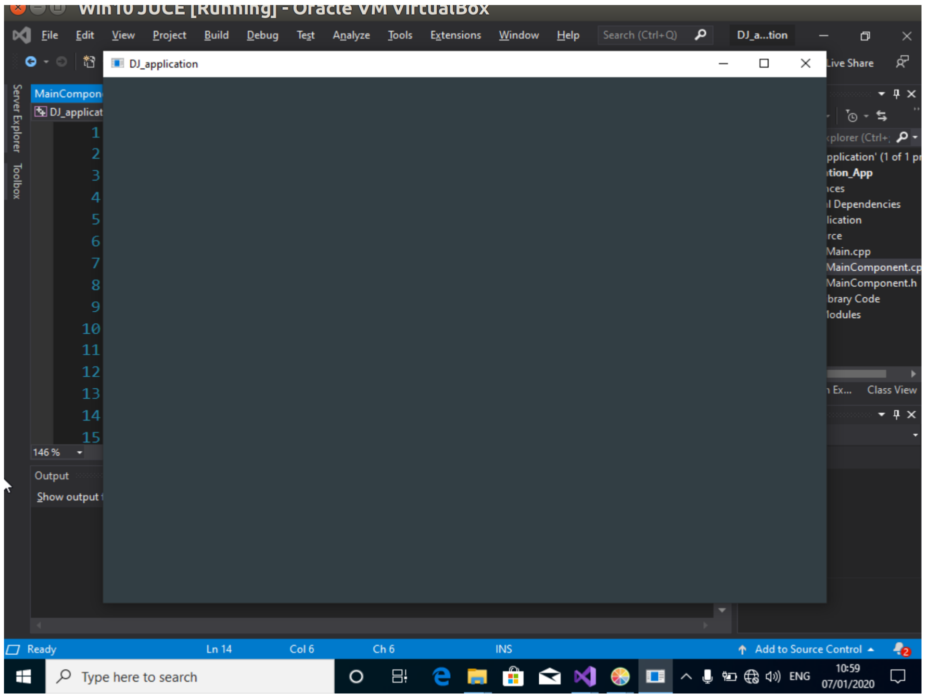
Your project should open in Visual Studio.

In the project explorer panel of Visual Studio, open up the DJ_application (if that is what you called it) folder then the Source folder, then open the MainComponent.cpp file. You should see something like this:



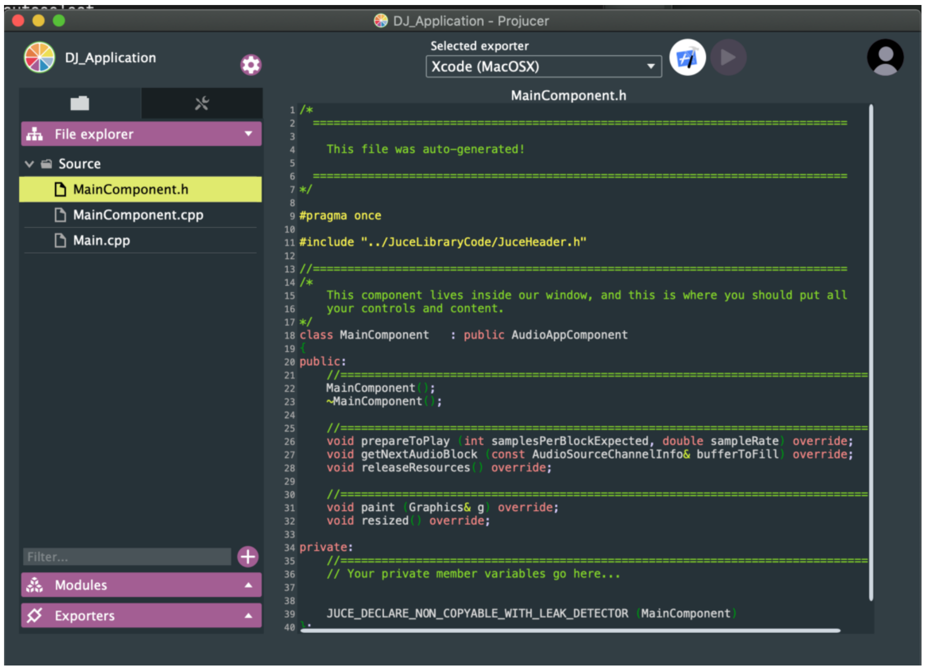
```
/*
  ======================================================

    This file was auto-generated!

  ======================================================
*/

#include "MainComponent.h"

//======================================================
MainComponent::MainComponent()
{
    // Make sure you set the size of the component aft
    // you add any child components.
    setSize (800, 600);

    // Some platforms require permissions to open inpu
    if (RuntimePermissions::isRequired (RuntimePermiss
        && ! RuntimePermissions::isGranted (RuntimePer
    {
        RuntimePermissions::request (RuntimePermission
                                     [&] (bool granted
```

- From the Visual Studio Build menu, select ‘Build Solution’.
- If everything goes well, from the Debug menu, select ‘Start without debugging’.
- You should see your new JUCE application running, like this:


Win10 JUCE [Running] - Oracle VM VirtualBox
File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
DJ_application
Ready
Ln 14
Col 6
Ch 6
INS
Add to Source Control
Type here to search

Mac

In Projucer, with your new project visible, as follows:

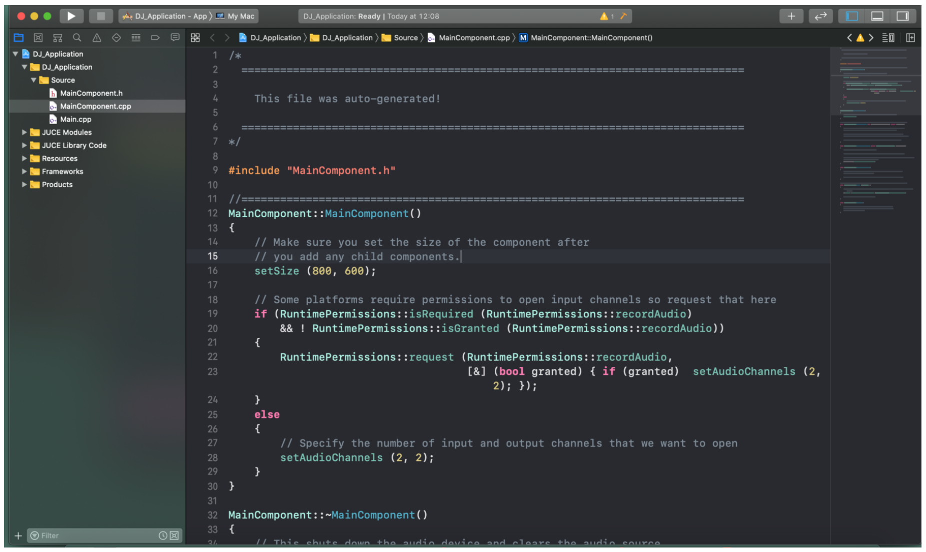


```
/*
  ==============================================================================

    This file was auto-generated!

  ==============================================================================
*/

#pragma once

#include "../JuceLibraryCode/JuceHeader.h"

//==============================================================================
/*
    This component lives inside our window, and this is where you should put all
    your controls and content.
*/
class MainComponent   : public AudioAppComponent
{
public:
    //==============================================================================
    MainComponent();
    ~MainComponent();

    //==============================================================================
    void prepareToPlay (int samplesPerBlockExpected, double sampleRate) override;
    void getNextAudioBlock (const AudioSourceChannelInfo& bufferToFill) override;
    void releaseResources() override;

    //==============================================================================
    void paint (Graphics& g) override;
    void resized() override;

private:
    //==============================================================================
    // Your private member variables go here...


    JUCE_DECLARE_NON_COPYABLE_WITH_LEAK_DETECTOR (MainComponent)
};
```

- Click the selected exporter menu at the top of the Projucer window and select Xcode (MacOSX).
- Click the Xcode icon next to the drop-down menu to open the project in Xcode.

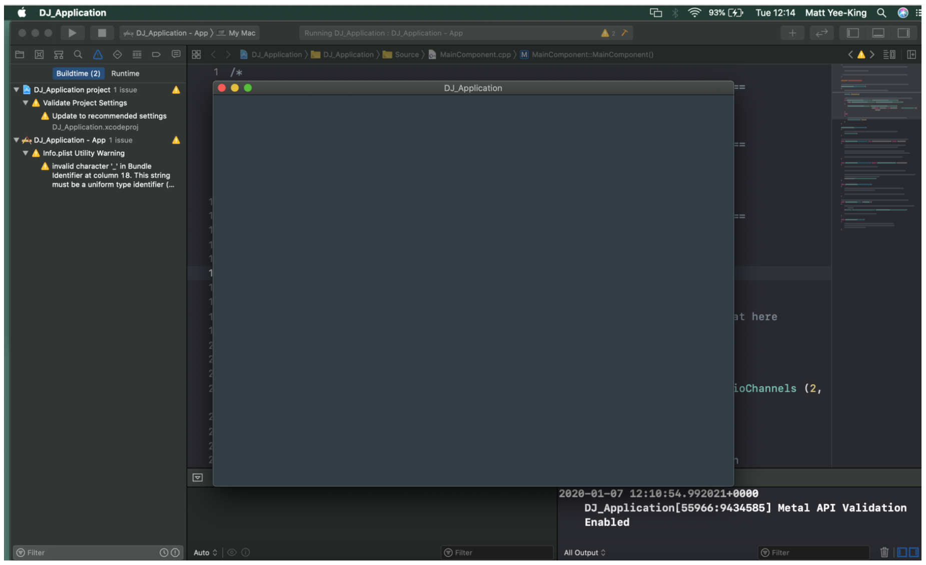
Your project should open in Xcode.

In the Project Navigator panel of Xcode (click on the folder icon visible in the left panel in the screenshot below), open up the DJ_application (if that is what you called it) folder then the Source folder, then open the MainComponent.cpp file. You should see something like this:



```
/*
  ==============================================================================

    This file was auto-generated!

  ==============================================================================
*/

#include "MainComponent.h"

//==============================================================================
MainComponent::MainComponent()
{
    // Make sure you set the size of the component after
    // you add any child components.
    setSize (800, 600);

    // Some platforms require permissions to open input channels so request that here
    if (RuntimePermissions::isRequired (RuntimePermissions::recordAudio)
        && ! RuntimePermissions::isGranted (RuntimePermissions::recordAudio))
    {
        RuntimePermissions::request (RuntimePermissions::recordAudio,
                                     [&] (bool granted) { if (granted)  setAudioChannels (2,
                                        2); });
    }
    else
    {
        // Specify the number of input and output channels that we want to open
        setAudioChannels (2, 2);
    }
}

MainComponent::~MainComponent()
{
```

- From the Xcode Product menu, select ‘Run’, or click the play button icon next to the window controls on the top left.
- If everything goes well, you should see your new JUCE application running, like this:



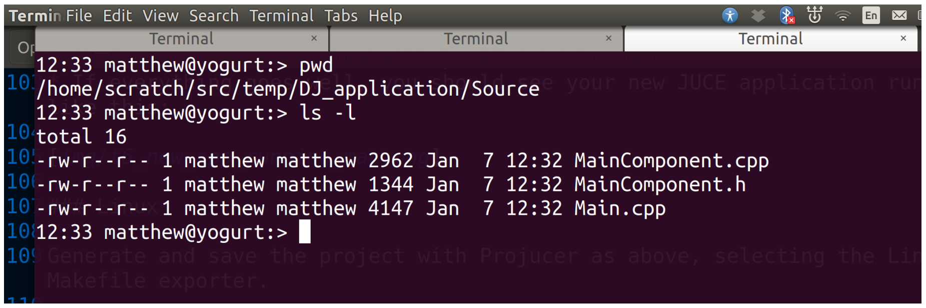
```
DJ_Application project 1 issue
  Validate Project Settings
    Update to recommended settings
    DJ_Application.xcodeproj
DJ_Application - App 1 issue
  Info.plist Utility Warning
    invalid character '_' in Bundle
    Identifier at column 18. This string
    must be a uniform type identifier (...
```

```
2020-01-07 12:10:54.992021+0000
    DJ_Application[55966:9434585] Metal API Validation
    Enabled
```

Linux

Generate and save the project with Projucer as above, selecting the Linux Makefile exporter.

In the terminal, cd into the Source folder of the project:


```
12:33 matthew@yogurt:> pwd
/home/scratch/src/temp/DJ_application/Source
12:33 matthew@yogurt:> ls -l
total 16
-rw-r--r-- 1 matthew matthew 2962 Jan  7 12:32 MainComponent.cpp
-rw-r--r-- 1 matthew matthew 1344 Jan  7 12:32 MainComponent.h
-rw-r--r-- 1 matthew matthew 4147 Jan  7 12:32 Main.cpp
12:33 matthew@yogurt:>
```

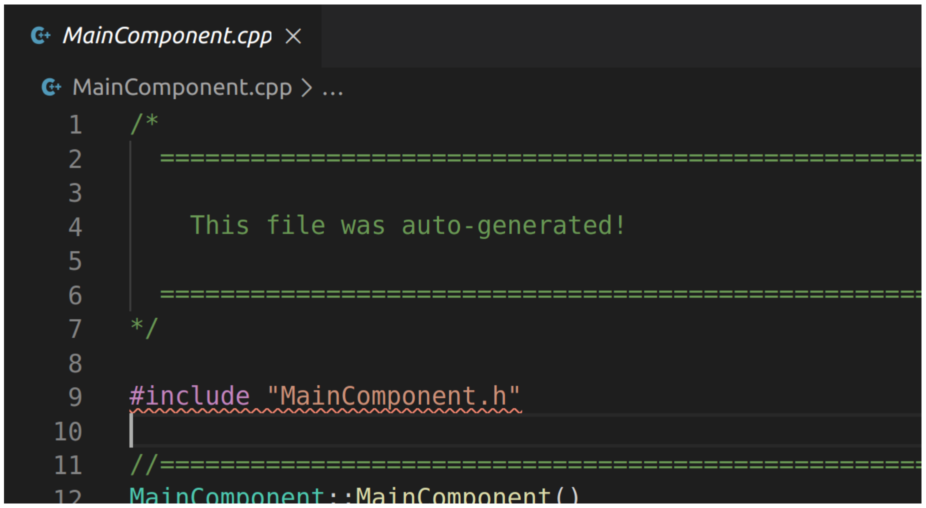
Create a .vscode folder

```
mkdir .vscode
```

Open the folder using Visual Studio Code

```
code ./
```

Open the Source folder and the MainComponent.cpp file. You should see something like this (I have moved my controls to the right-hand side)



```
/*
  ==========================================================

    This file was auto-generated!

  ==========================================================
*/

#include "MainComponent.h"

//==========================================================
MainComponent::MainComponent()
```

Right-click on the .vscode folder, and add a new file called 'c_cpp_properties.json'. Put this into the file:

```
{
    "configurations": [
        {
            "name": "Linux",
            "includePath": [
                "${workspaceFolder}/**",
                "/home/scratch/src/juce/JUCE/modules/"
            ],

            "browse": {
                "limitSymbolsToIncludedHeaders": false,
                "path": [
                    "${workspaceFolder}/**",
                    "/home/scratch/src/juce/JUCE/modules/**"
                ]
            },

            "defines": [],
            "compilerPath": "/usr/bin/clang-6.0",
            "cStandard": "c11",
            "cppStandard": "c++17",
            "intelliSenseMode": "clang-x64"
        }
    ],
    "version": 4
}
```

Change '/home/scratch/src/juce/JUCE/modules/' to point to your JUCE folder. That should set up your CPP environment so it can auto-complete and so on.

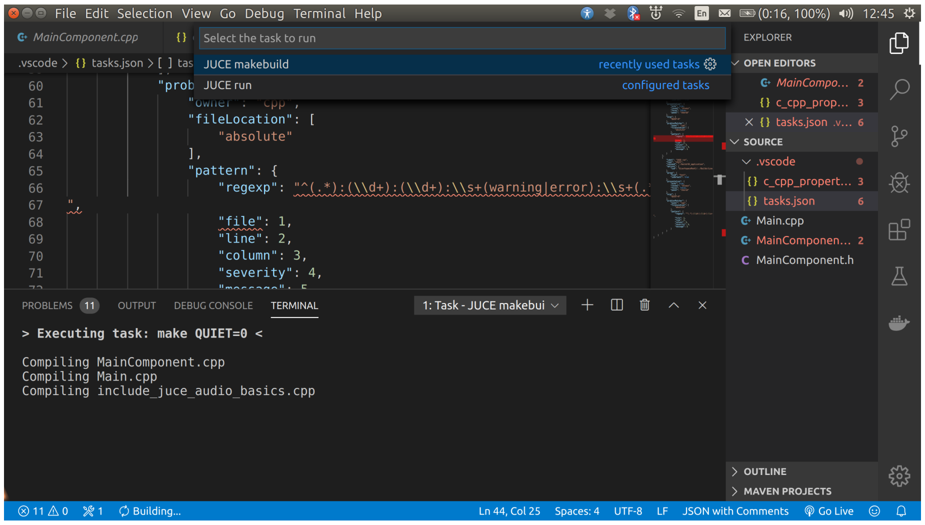
Now create another new file in the .vscode folder called 'tasks.json'. Put this into that file:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "JUCE makebuild",
            "type": "shell",
            "command": "make",
            "options": {
                "cwd": "${workspaceRoot}/../Builds/LinuxMakefile/"
            },
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "presentation": {
                "echo": true,
                "reveal": "always",
                "focus": false,
                "panel": "shared"
            },
            "args": [
                "QUIET=0"
            ],
            "problemMatcher": {
                "owner": "cpp",
                "fileLocation": [
                    "absolute"
                ],
                "pattern": {
                    "regexp": "^(.*):(\\d+):(\\d+):\\s+(warning|error):\\s+(.*)$
",
                    "file": 1,
                    "line": 2,
                    "column": 3,
                    "severity": 4,
                    "message": 5
                }
            }
        },
        {
            "label": "JUCE run",
            "type": "shell",
            "command": "./build/DJ_Application",
            "options": {
                "cwd": "${workspaceRoot}/../Builds/LinuxMakefile/"
            },
            "group": {
                "kind": "test",
                "isDefault": true
            },
            "presentation": {
                "echo": true,
                "reveal": "always",
                "focus": false,
                "panel": "shared"
            },
            "args": [
                "QUIET=0"
            ],
            "problemMatcher": {
                "owner": "cpp",
                "fileLocation": [
                    "absolute"
                ],
                "pattern": {
                    "regexp": "^(.*):(\\d+):(\\d+):\\s+(warning|error):\\s+(.*)$
",
          "file": 1,
          "line": 2,
          "column": 3,
          "severity": 4,
          "message": 5
        }
      }
    }
  ]
}
```

Change ./build/DJ_Application to the name of your application if it is different from DJ_Application and save.

Now you should be able to build and run your application.

To build, go to the Terminal menu in Visual Studio Code and select Run Task:

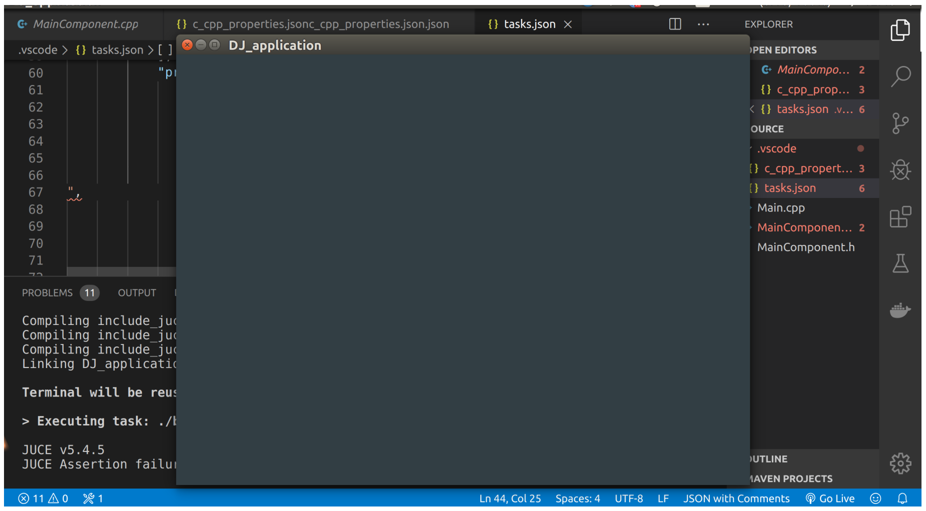


```
                "prob
                    "owner": "cpp",
                    "fileLocation": [
                        "absolute"
                    ],
                    "pattern": {
                        "regexp": "^(.*):(\\d+):(\\d+):\\s+(warning|error):\\s+(.
",
                        "file": 1,
                        "line": 2,
                        "column": 3,
                        "severity": 4,
```

```
> Executing task: make QUIET=0 <

Compiling MainComponent.cpp
Compiling Main.cpp
Compiling include_juce_audio_basics.cpp
```

You should see the tasks from your tasks.json file. Select ‘JUCE makebuild’. This will build using the makefile.

If there are no errors, select Run Task again and select the ‘JUCE run’. You should see your app running like this:



```
Compiling include_juc
Compiling include_juc
Compiling include_juc
Linking DJ_applicatio

Terminal will be reus

> Executing task: ./b

JUCE v5.4.5
JUCE Assertion failu
```

Troubleshooting

When creating projects, sometimes JUCE cannot find the modules folder. JUCE modules are the components of the JUCE library that you use to build your application, for example, GUI widgets and audio handling components. The modules are stored in a modules folder in your JUCE folder (the one you get when you unpack the JUCE library download). You need to point Projucer at this folder.