Материалы к занятию

Вы написали свою игру с помощью Arcade. Теперь вы хотите поделиться ей с другими. Обычно это означает помощь людям в установке Python, загрузку необходимых модулей, копирование вашего кода, а затем заставить все это работать. Но есть более удобное решение - модуль PyInstaller!
PyInstaller - это инструмент для Python, который позволяет объединить все приложение Python в исполняемый пакет из одного файла, которым вы можете легко поделиться.

Разберем по шагам:

- Установка PyInstaller
- Создание простого примера приложения, использующего Arcade
- Объединение приложения в однофайловый исполняемый файл
- Запуск приложения

Во-первых, установим модуль
pip install arcade pyinstaller

Для примера создадим простое окно. Нам нужна однофайловая игра, которая не требует никаких дополнительных изображений или звуков. Далее мы можем усложнить пример.
Создадим файл

```
import arcade

SCREEN_WIDTH = 400
SCREEN_HEIGHT = 400
SCREEN_TITLE = "Demo"


class MyGame(arcade.Window):
   def __init__(self):
       super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT,
SCREEN_TITLE)

arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)

   def on_draw(self):
       self.clear()
       arcade.draw_circle_filled(200, 200, 100,
arcade.color.BLUE)
       arcade.finish_render()


def main():
   window = MyGame()
   arcade.run()
```

```
if __name__ == "__main__":
   main()
```

Теперь создадим файл, запустив PyInstaller из командной строки:
pyinstaller main.py --onefile

PyInstaller создает исполняемый файл, который является пакетом вашей игры. Он помещает его в папку с текущим рабочим каталогом.
Вы можете скопировать этот файл в любое место на вашем компьютере и запустить его. Или поделитесь им с с кем то еще. Все, что нужно вашему скрипту, находится внутри этого исполняемого файла.
Для простых игр это все, что вам нужно знать! Но, если ваша игра загружает какие-либо файлы данных с диска, то требуется немного больше.

При создании файла PyInstaller сначала проверяет ваш проект и автоматически идентифицирует почти все, что нужно вашему проекту (интерпретатор Python, установленные модули и т. д.). Но он не может автоматически определить, какие файлы ваша игра загружает с диска (изображения, звуки, карты). Таким образом, вы должны явно сообщить PyInstaller об этих файлах и о том, куда он должен поместить их в пакет. Это делается с помощью флага PyInstaller:
pyinstaller main.py --add-data "stripes.jpg;."

После указания PyInstaller включить файлы данных в пакет необходимо убедиться, что код загружает файлы данных из правильного каталога. Когда вы делитесь проектом, вы не можете контролировать, из какого каталога пользователь будет запускать ваш пакет. Это осложняется тем, что однофайловый пакет PyInstaller распаковывается во время выполнения в случайный временный каталог, а затем выполняется оттуда. Рассмотрим один простой подход, который позволяет коду выполнять и загружать файлы при запуске в пакете PyInstaller, а также иметь возможность запускаться, когда он не входит в комплект поставки. Требуется сделать две вещи. Во-первых, приведенный ниже фрагмент должен быть размещен в начале кода:

```
if getattr(sys, 'frozen', False) and hasattr(sys, '_MEIPASS'):
    os.chdir(sys._MEIPASS)
```

Если у вас просто есть один файл данных в том же каталоге, что и ваш скрипт, требуется обратиться к файлу данных, используя относительный путь:
sprite = arcade.Sprite("stripes.jpg")

Затем нужно использовать команду PyInstaller, чтобы включить файл данных в исполняемый файл:
pyinstaller main.py --add-data "stripes.jpg;."
или для всех файлов jpg
pyinstaller main.py --add-data "stripes.jpg;."

Если ресурсы игры лежат в директории рядом со скриптом, то необходимо обратиться к ним таким образом
sprite = arcade.Sprite("images/player.jpg")
sprite = arcade.Sprite("images/enemy.jpg")

Затем указываем команду, чтобы каталог тоже попал в исполняемый файл
pyinstaller main.py --add-data "images;images"

Флаги можно добавлять множество раз

pyinstaller main.py --add-data "player.jpg;." --add-data "enemy.jpg;." --add-data "music;music"

Также можно создать исполняемый файл с помощью еще одного стороннего модуля.
Установим Nuitka:
pip install nuitka

Использовать можно код из прототипа платформера
Преобразовать этот код в автономный исполняемый файл можно с помощью команды:
python -m nuitka main.py --standalone --enable-plugin=numpy

Компиляция занимает время, иногда может быть до 2 часов, в зависимости от характеристик вашего ПК. После завершения процесса появятся две новые папки с именем. Папка build нас не интересует на данный момент. Просто зайдите в папку dist и запустите файл, присутствующий там. Если ошибок нет, то приложение должно работать

Примечание: Если вы хотите скомпилировать код в один файл вместо папки, просто снимите флаг standalone и добавьте флаг onefile

Когда приложение использует пользовательские файлы данных, которые могут включать звуковые эффекты, шрифты и т. д., чтобы связать их с приложением, надо указать флаг
python -m nuitka main.py --standalone --enable-plugin=numpy
--include-data-file=C:/Users/Hunter/Desktop/my_game/my_image.png

При этом файл с именем my_image.png в указанном расположении будет скопирован в корень исполняемого файла/
Чтобы объединить целую папку:

python -m nuitka main.py --standalone --enable-plugin=numpy
--include-data-dir=C:/Users/Hunter/Desktop/my_game/assets

При этом вся папка assets, названная в указанном расположении, будет скопирована в корневой каталог исполняемого файла.

Возможно, вы заметили, что при открытии исполняемого файла автоматически открывается окно консоли. Несмотря на то, что оно полезно при отладке и ошибках, оно выглядит некрасиво. Можно убрать вывод консоли с помощью команды:
python -m nuitka main.py --standalone --windows-force-stderr-spec=%PROGRAM%logs.txt
--windows-force-stdout-spec=%PROGRAM%output.txt

Nuitka предоставляет нам флаги (варьируется для каждой ОС) для установки пользовательских значков. Первый флаг принимает файл и устанавливает его в качестве значка приложения:
python -m nuitka 17_views.py --standalone --windows-icon-from-ico=icon.png

При этом для значка приложения будет установлено значение icon.png
python -m nuitka 17_views.py --standalone
--windows-icon-from-exe=C:\Users\Hunter\AppData\Local\Programs\Python\Python310/python.exe

Это установит значок приложения на значок Python

Вы написали свою игру с помощью Arcade. Теперь вы хотите поделиться ей с другими. Обычно это означает помощь людям в установке Python, загрузку необходимых модулей, копирование вашего кода, а затем заставить все это работать. Но есть более удобное решение - модуль PyInstaller!
PyInstaller - это инструмент для Python, который позволяет объединить все приложение Python в исполняемый пакет из одного файла, которым вы можете легко поделиться.

Разберем по шагам:

- Установка PyInstaller
- Создание простого примера приложения, использующего Arcade
- Объединение приложения в однофайловый исполняемый файл
- Запуск приложения

Во-первых, установим модуль
pip install arcade pyinstaller

Для примера создадим простое окно. Нам нужна однофайловая игра, которая не требует никаких дополнительных изображений или звуков. Далее мы можем усложнить пример. Создадим файл

```
import arcade

SCREEN_WIDTH = 400
SCREEN_HEIGHT = 400
SCREEN_TITLE = "Demo"


class MyGame(arcade.Window):
   def __init__(self):
       super().__init__(SCREEN_WIDTH, SCREEN_HEIGHT,
SCREEN_TITLE)

arcade.set_background_color(arcade.csscolor.CORNFLOWER_BLUE)

   def on_draw(self):
       self.clear()
       arcade.draw_circle_filled(200, 200, 100,
arcade.color.BLUE)
       arcade.finish_render()


def main():
   window = MyGame()
   arcade.run()


if __name__ == "__main__":
   main()
```

Теперь создадим файл, запустив PyInstaller из командной строки:
pyinstaller main.py --onefile

PyInstaller создает исполняемый файл, который является пакетом вашей игры. Он помещает его в папку с текущим рабочим каталогом.
Вы можете скопировать этот файл в любое место на вашем компьютере и запустить его. Или поделитесь им с с кем то еще. Все, что нужно вашему скрипту, находится внутри этого исполняемого файла.
Для простых игр это все, что вам нужно знать! Но, если ваша игра загружает какие-либо файлы данных с диска, то требуется немного больше.

При создании файла PyInstaller сначала проверяет ваш проект и автоматически идентифицирует почти все, что нужно вашему проекту (интерпретатор Python, установленные модули и т. д.). Но он не может автоматически определить, какие файлы ваша игра загружает с диска (изображения, звуки, карты). Таким образом, вы должны явно

сообщить PyInstaller об этих файлах и о том, куда он должен поместить их в пакет. Это делается с помощью флага PyInstaller:
pyinstaller main.py --add-data "stripes.jpg;."

После указания PyInstaller включить файлы данных в пакет необходимо убедиться, что код загружает файлы данных из правильного каталога. Когда вы делитесь проектом, вы не можете контролировать, из какого каталога пользователь будет запускать ваш пакет. Это осложняется тем, что однофайловый пакет PyInstaller распаковывается во время выполнения в случайный временный каталог, а затем выполняется оттуда. Рассмотрим один простой подход, который позволяет коду выполнять и загружать файлы при запуске в пакете PyInstaller, а также иметь возможность запускаться, когда он не входит в комплект поставки. Требуется сделать две вещи. Во-первых, приведенный ниже фрагмент должен быть размещен в начале кода:

```
if getattr(sys, 'frozen', False) and hasattr(sys, '_MEIPASS'):
    os.chdir(sys._MEIPASS)
```

Если у вас просто есть один файл данных в том же каталоге, что и ваш скрипт, требуется обратиться к файлу данных, используя относительный путь:
sprite = arcade.Sprite("stripes.jpg")

Затем нужно использовать команду PyInstaller, чтобы включить файл данных в исполняемый файл:
pyinstaller main.py --add-data "stripes.jpg;."
или для всех файлов jpg
pyinstaller main.py --add-data "stripes.jpg;."

Если ресурсы игры лежат в директории рядом со скриптом, то необходимо обратиться к ним таким образом
sprite = arcade.Sprite("images/player.jpg")
sprite = arcade.Sprite("images/enemy.jpg")

Затем указываем команду, чтобы каталог тоже попал в исполняемый файл
pyinstaller main.py --add-data "images;images"

Флаги можно добавлять множество раз

pyinstaller main.py --add-data "player.jpg;." --add-data "enemy.jpg;." --add-data "music;music"

Также можно создать исполняемый файл с помощью еще одного стороннего модуля. Установим Nuitka:

pip install nuitka

Использовать можно код из прототипа платформера
Преобразовать этот код в автономный исполняемый файл можно с помощью команды:
python -m nuitka main.py --standalone --enable-plugin=numpy

Компиляция занимает время, иногда может быть до 2 часов, в зависимости от характеристик вашего ПК. После завершения процесса появятся две новые папки с именем. Папка build нас не интересует на данный момент. Просто зайдите в папку dist и запустите файл, присутствующий там. Если ошибок нет, то приложение должно работать

Примечание: Если вы хотите скомпилировать код в один файл вместо папки, просто снимите флаг standalone и добавьте флаг onefile

Когда приложение использует пользовательские файлы данных, которые могут включать звуковые эффекты, шрифты и т. д., чтобы связать их с приложением, надо указать флаг
python -m nuitka main.py --standalone --enable-plugin=numpy
--include-data-file=C:/Users/Hunter/Desktop/my_game/my_image.png

При этом файл с именем my_image.png в указанном расположении будет скопирован в корень исполняемого файла/
Чтобы объединить целую папку:
python -m nuitka main.py --standalone --enable-plugin=numpy
--include-data-dir=C:/Users/Hunter/Desktop/my_game/assets

При этом вся папка assets, названная в указанном расположении, будет скопирована в корневой каталог исполняемого файла.

Возможно, вы заметили, что при открытии исполняемого файла автоматически открывается окно консоли. Несмотря на то, что оно полезно при отладке и ошибках, оно выглядит некрасиво. Можно убрать вывод консоли с помощью команды:
python -m nuitka main.py --standalone --windows-force-stderr-spec=%PROGRAM%logs.txt
--windows-force-stdout-spec=%PROGRAM%output.txt

Nuitka предоставляет нам флаги (варьируется для каждой ОС) для установки пользовательских значков. Первый флаг принимает файл и устанавливает его в качестве значка приложения:
python -m nuitka 17_views.py --standalone --windows-icon-from-ico=icon.png

При этом для значка приложения будет установлено значение icon.png
python -m nuitka 17_views.py --standalone
--windows-icon-from-exe=C:\Users\Hunter\AppData\Local\Programs\Python\Python310/python.exe

Это установит значок приложения на значок Python