

IAE 101 – Introduction to Digital Intelligence
Fall, 2024
Programming Project 01 – Musical Dice Game

Assigned: Friday, September 27th, 2024

Due: **Thursday, October 10th, 2024, at 11:59 PM**

Description:¹

In 1787, Wolfgang Amadeus Mozart created a dice game ([Mozart's Musikalisches Würfelspiel](#)). In the game, you compose a two-part waltz by pasting together 32 of 272 pre-composed musical elements at random.

The waltz. The waltz consists of two parts - the minuet and the trio. Each is comprised of 16 measures, which are generated at random according to a fixed set of rules, as described below.

- *Minuet.* The minuet consists of 16 measures. There are 176 possible Minuet measures, named M1.wav through M176.wav. To determine which one to play, roll two fair dice for each column of the following table.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	96	22	141	41	105	122	11	30	70	121	26	9	112	49	109	14
3	32	6	128	63	146	46	134	81	117	39	126	56	174	18	116	83
4	69	95	158	13	153	55	110	24	66	139	15	132	73	58	145	79
5	40	17	113	85	161	2	159	100	90	176	7	34	67	160	52	170
6	148	74	163	45	80	97	36	107	25	143	64	125	76	136	1	93
7	104	157	27	167	154	68	118	91	138	71	150	29	101	162	23	151
8	152	60	171	53	99	133	21	127	16	155	57	175	43	168	89	172
9	119	84	114	50	140	86	169	94	120	88	48	166	51	115	72	111
10	98	142	42	156	75	129	62	123	65	77	19	82	137	38	149	8
11	3	87	165	61	135	47	147	33	102	4	31	164	144	59	173	78
12	54	130	10	103	28	37	106	5	35	20	108	92	12	124	44	131

For example, if you roll an 11 for measure 3, then play measure [165](#).

- *Trio.* The trio consists of 16 measures. There are 96 possible Trio measures named T1.wav through T96.wav. To determine which one to play, roll one fair die for each column, and use the following table.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	72	6	59	25	81	41	89	13	36	5	46	79	30	95	19	66
2	56	82	42	74	14	7	26	71	76	20	64	84	8	35	47	88
3	75	39	54	1	65	43	15	80	9	34	93	48	69	58	90	21
4	40	73	16	68	29	55	2	61	22	67	49	77	57	87	33	10
5	83	3	28	53	37	17	44	70	63	85	32	96	12	23	50	91
6	18	45	62	38	4	27	52	94	11	92	24	86	51	60	78	31

¹ This description is stolen, wholesale, from <https://introcs.cs.princeton.edu/java/assignments/mozart.html>

There are $11^{16} * 6^{16}$ different possible results, some of which are more likely than others. Since this is over 10^{23} different possibilities, each time you play the game you are likely to compose a piece of music that has never been heard before! Mozart carefully constructed the measures to obey a rigid harmonic structure, so each waltz reflects Mozart's distinct style. Unfortunately, due to the rigidity, the process never results in anything truly extraordinary.

You will implement a program that constructs and plays a random waltz according to the rules of Mozart's game.

Directions:

Two setup steps must be completed before beginning the assignment proper.

First, please install the `simpleaudio-patched` Python package. This can be done with Python's Package Installer, **pip**. On Windows open the command prompt CMD (or Powershell), on MAC start the terminal, and enter the following commands (Note: YOU CANNOT DO THIS IN IDLE):

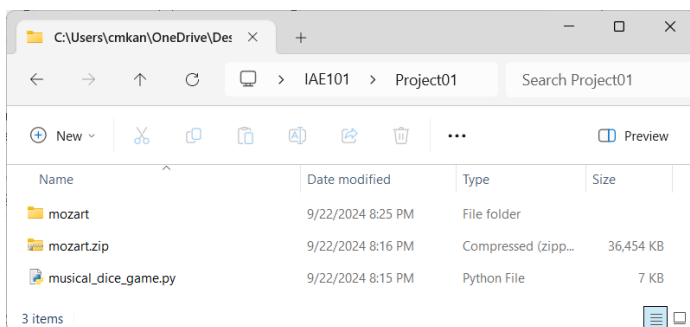
Windows: `python -m pip install simpleaudio-patched --user`
(You may need to use 'py -3' in place of 'python')

Mac: `python -m pip install simpleaudio-patched --user`
(You may need to use 'python3' in place of python)

Please try to do the installation as soon as possible (this weekend) so that any installation problems can be resolved in a timely fashion.

Second, the audio files are provided to you as a ZIP archive. You must unzip the archive in order to access the files. Download the zip archive. Decompress it--your computer should know how to do this--**but**, just double-clicking the file won't be enough.

Make sure that the assignment file, **musical_dice_game.py**, is in the same folder as the **mozart** folder that contains the music files. Your setup should look like this:



Now you are ready to begin the assignment. For this assignment, you will complete the implementation of **musical_dice_game.py**.

This file contains two tables represented as nested lists. The first table contains Mozart's table for randomly choosing 16 measures for the minuet portion of the waltz. The second table contains Mozart's table for randomly choosing 16 measures for the trio portion of the waltz. The values in each table are numbers of measures of music represented as strings. Each of these numbers uniquely identifies an audio file distributed with the assignment—the number is a part of the audio file's name.

You are responsible for implementing four functions. The first function, `minuet_filename()`, takes one argument, which is a string. That string will be the number of one of the entries randomly selected from Mozart's minuet table. The function will return a string that is the filepath for the corresponding audio file. Each minuet musical file begins with a capital "M", and then the number from the table, and then a period, ".", and then the file extension for the audio file, "wav". The files are located in the `mozart` directory, so that will be included as well.

The string should be composed like this: `"mozart/M<minuet number>.wav"`

For example, if you select "119" by rolling a 9 for the first minuet measure, then the corresponding filename will be the string `"mozart/M119.wav"`.

The second function, `trio_filename()`, is very similar. It also takes one argument, a numeral represented as a string that corresponds to the name of one of the audio files that contains a measure of trio music. Again, the function must return the complete filepath of the audio file as a string. The filename begins with a capital "T", followed by the number from the table, then a period, ".", and then the file extension for the audio file, "wav". The trio files are also located in the `mozart` folder, so that must be included to complete the filepath.

The string should be composed like this: `"mozart/T<trio number>.wav"`

For example, if you select "49" by rolling a 4 for the 11th trio measure, then the corresponding filename will be the string `"mozart/T49.wav"`.

The third function, `roll_dice()`, is used to simulate the dice rolling. The function takes one integer argument that specifies how many 6-sided dice will be rolled. You will use functions from Python's `random` library to simulate rolling the specified number of 6-sided dice. The function will then return the sum of all the dice rolls performed by the function.

For example, if the first "die roll" produced a 5 and the second "die roll" produces a 2, then the result returned by the `roll_dice()` function should be 7.

The final function is called `construct_waltz()`. It does not take any arguments, and it does not return a value. Instead, you will loop over the columns of the minuet table and use the function call `roll_dice(2)` to simulate a roll of two 6-sided dice to randomly select a number identifying a musical measure from each column. You must record your selections. I recommend appending each selection to a list.

The function will then perform another, similar loop over the trio table, using the function call `roll_dice(1)` to simulate rolling one 6-sided die to randomly select a number identifying a musical measure from each column and appending the result to your list of selections.

After you have randomly selected all the measures that will make up your waltz, you then need to play your waltz, so you need to access the audio files that will compose your waltz for playback.

Loop over your list of selections and use the `minuet_filename()` and `trio_filename()` functions to construct filenames for the corresponding .wav audio files. Then pass each filename to the `simpleaudio.WaveObject.from_wave_file()` in order to construct an object of the `simpleaudio.WaveObject` class. You should append each of these objects to a new list as you create them.

Now loop over your list of `WaveObjects` and call the `play()` method on each one. Make sure that you wait until each measure has finished playing before playing the next one.

Bonus (5 points): You can earn 5 bonus points for using the `play_buffer()` function to play your waltz rather than the `play()` function. (You can only earn the bonus credit if it works—you may want to implement your solution using the `play()` function first before then trying to use `play_buffer()` instead.)

And that is it. Your random waltz generator is now complete.

Submission Instructions:

1. Please add your name, netid, and student ID to the top of `musical_dice_game.py` as comments.
2. Upload your `musical_dice_game.py` file to Blackboard.