DOING TASKS CONCURRENTLY

1. Objectives

The main goal of this assignment is:

- To learn how to create and start a thread to perform a certain task, suspend and resume the task at any time.
- How to let several threads do their tasks simultaneously and operate without interference.

2. DESCRIPTION

Your job in this first assignment is to write a simple GUI-based application that performs different tasks at the same time. You are given a number tasks of which you should pick-up at least 2 tasks, design a graphical user interface (GUI) and create an application to run the tasks concurrently. The tasks are to be executed each in its own thread. This way, you application will be running with at least three threads: the main thread (GUI) that is a part of the process (your program) and the two news threads that you will be creating for your tasks. You can, of course, select more than two tasks if you wish to practice with more threads, but to qualify for a **Pass** grade, you only need to use **two threads** among the following alternatives (you can of course use more alternatives optionally):

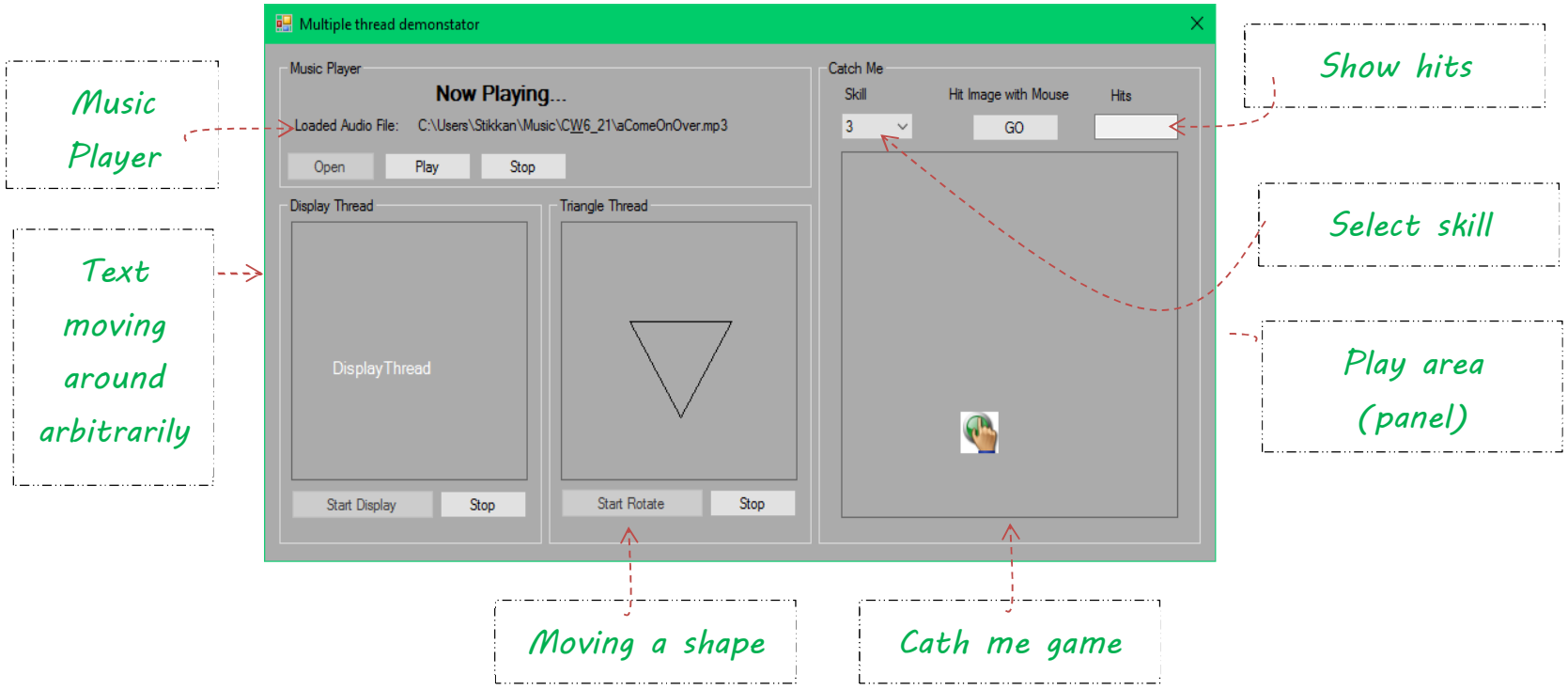
- Task 1: Displaying text randomly on a panel (see figure).
- Task 2: Turning a graphical object (see figure).
- Task 3: Playing a music file (see figure)
- Task 4: Playing a simple game: Catch Me (see figure)

The tasks that you choose should run in separate threads but simultaneously, independent of each other. You may use any of the languages C#, Java or C++ (can use Visual C++). For C#, it is recommended to use **Visual Studio** to design your GUI and do the rest of coding. Visual Studios recent version is available for download for installation to your own computer from Microsoft Imagine (formerly called DreamSpark). The code for drawing the GUI will be provided so you can concentrate on your multithreading programming. However, you

may certainly program the GUI by yourself. It is recommended to use Visual Studio (C#) or **InilliJ** (Java) which make our grading job easier. We will be using Java 12

3. TASK 1 - DISPLAYING TEXT RANDOMLY

3.1 As illustrated in the left-panel of the image here, a separate thread is used to display a text (the thread's name in the example) appearing randomly on the drawing area, as demonstrated in the image below.



4. Task 2 - Moving a graphical object

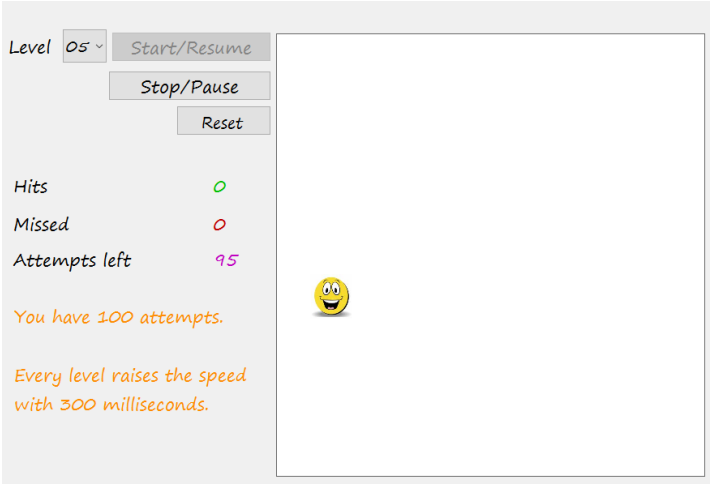
4.1 Create a thread to draw, move and/or rotate a shape. Draw any shape, a rectangle, a polygon, a curve or a triangle (as in the above example). The shape moves, rotates around its midpoint or bounce against the edges of the panel which contains the drawing. Another example is to draw a clock (digital or analogue) showing the current time (system time or any local time). You may even load (embed) an image file and move the image around inside the panel. The main idea is that something happens in the panel dynamically.

4.2 When the button **Start Rotate** is pressed, a thread is created and it starts drawing the shape on the panel. The panel is updated with an interval of a certain number of milliseconds (for example 200), and rotating a certain degree (say 10) around its midpoint, clockwise or counter-clockwise. If you choose to display a clock, it should update itself and display the time every second (Thread.Sleep(1000)).

4.3 There should be a **Start** and a **Stop** button for starting and stopping the thread working on each panel. The Start button should create and start the thread and the Stop button should end the thread execution. Make the buttons enabled only one at a time, depending on status. Can't start something that is already running or vice versa.

5. Task 3 - Music Player

5.1 Create a thread to play a music file. Use a file dialog to let the user select a music file to play. Open the music file using the computers default media player.



6. Catch-Me Game

6.1 The idea behind the Catch-Me game is to let an image or an icon appear randomly on a panel, at random intervals (shorter for higher levels). The user tries to hit the image by the mouse. The program displays the number

of hits on the figure, number of times missing the figure and the number of attempts left. Design your game using your own fantasy!

Hint: Write one listener for the image-click and one for the panel-click. The number of image-clicks = hits, panel-clicks = miss!

6.2 Disable the start button during a play session. At some intervals the play thread then randomly places an image on the playing panel. When the user clicks the GO button, make sure to wait one or two seconds before first image is shown so the user gets prepared.

7. Specifications and Requirements for a Pass grade (G)

7.1 It is not allowed to use a Timer control or any similar components. All such operations should be accomplished by the threads.

7.2 Do not use **Thread.Interuppt** or **Thread.Suspend** to stop a thread. Use a variable as a flag for each thread (IsT1Running) and set it to true while it is running, set to false to stop running.

7.3 Select two (or more) tasks and combine them as parts of your application.

7.4 Your code must be well organized into classes and well object-oriented based on Encapsulation (private instance variables, short methods), inheritance and polymorphism wherever and whenever applicable. Add comments to your code to document your thoughts.

7.5 Comment your code, briefly what the methods do and inside the code wherever needed..

7.6 Make sure that no user action can cause the program to crash. Test your application carefully before submitting.

7.7 Make sure to test that at no thread is active when closing the application; if so is the case, terminate before closing (can handle the **FormClosing** event if you use C#). You may also use **Thread.IsAlive** or **Thread.Join**.

7.8 Use one code file for one class without regard to the size of the class.

7.9 Use proper variable and methods names, a, fnc, and other such short names make you code difficult to read and maintain.

Hint: Do not use nested classes; they make your coding more complicated.

8. GRADING AND SUBMISSION

Compress all the files, folders and subfolders into a **zip**, **rar, 7z** file, and then upload it via the Assignment page on Canvas. Make sure that you submit the correct version of your project and that you have compiled and tested your project before handing in. Be careful not to use any hard-coded file paths (for example path to an image file on your C-drive) in your source code. It will not work on other computers. Projects that do not compile and run correctly, or is done with poor code quality, will be returned for completion and resubmission.

After or before submitting your assignment to the module, show your assignment to your lab leader during the scheduled hours in the labs.

9. LINKS:

Playing sound (C#): http://www.caveofprogramming.com/guest-articles/csharp/c-for-beginners-make-your-own-mp3-player-free/
Drawing 2D (C#): https://www.youtube.com/watch?v=R8KTm4E3gEA , https://www.youtube.com/watch?v=R8KTm4E3gEA

Drawing 2D (Java): https://www.youtube.com/watch?v=ydQWhluoBXM
Java threads: https://www.youtube.com/watch?v=br_TEuE8TbY. https://www.youtube.com/watch?v=F-CkaU8aQZI

Threads (C++): http://www.tutorialspoint.com/cplusplus/cpp_multithreading.htm

Good Luck!

Farid Naisan,
Course Responsible and Instructor