This week, we are going to develop a simple game that allows the user to select the correct numbers for every denomination for a change.

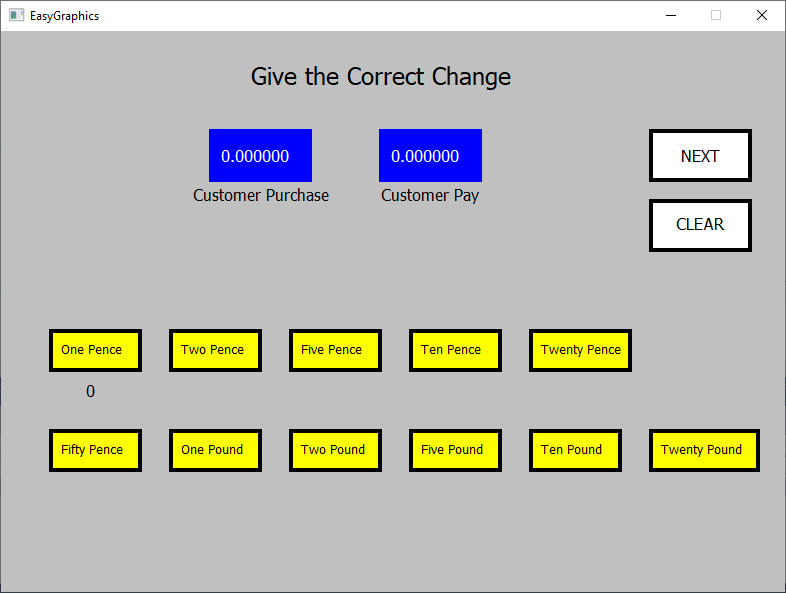
The denominations that should be available in your application are,

**1 pence 2 pence 5 pence 10 pence 20 pence 50 pence**

**1 pound 2 pound 5 pound 20 pound**

The user will be given 10 sets of purchase and rendered amounts (randomly). The user needs to give the correct change by clicking the denominations.

Your task this week is to design a GUI for the application using Easy Graphics and make some of your GUI components function. You may design the GUI similar to this (You should design a better GUI than this). Add more components in your GUI such as player’s name, mark, labels for correct or incorrect.



Let design a dummy GUI first.

**Adding Pictures**

What if you would like to use pictures instead of rectangles, triangles, circles and texts? Follow these steps,

1. Go to your project folder explorer (“Open Folder in File Explorer) – Refer lesson 1.
2. Create a folder and name the folder to your preferred name.
3. Add your pictures to the folder. The pictures format should be **.bmp**.
4. Use this code to put the picture in your canvas.

drawBitmap(L"assets\\dogface.bmp", 200, 200, 100, 250);

Adjust the values for the positions and size.

The picture file.

The folder you created

Now replace the rectangles to pictures!

**Creating a Random Number Function for Purchase**

Now we are going to write a void function to get an amount for purchase. We also would like to store the value in an array. We can create an array that stores the amount for purchase and customer pay.

Firstly, create a global variable in ***DrawingTool.h*** file.

double randomPurchase;

Then, declare an array (of size 2) with default values 0.00.

double amounts[2] = { 0.00,0.00 }; // index 0 is for purchase amount, index 1 is for pay amount

Now write the following code in ***DrawingTool.cpp*** file. You may write your own way how to get a random decimal number.

double DrawingTool::getRandomPurchase()

{

srand((double)time(NULL));

double decimal = (double)rand()/(RAND\_MAX);

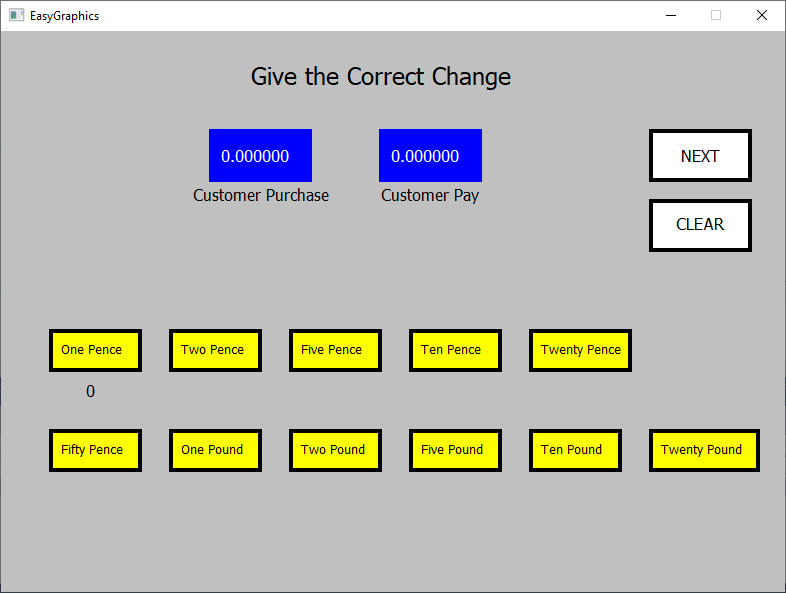
int a = rand() % 100;

decimal += (double)a;

return randomPurchase = decimal;

}

You need to include #include <time.h> in the file!



We will look into calling the “getRandomPurchase”

function from the “NEXT” rectangle and set the value to “Customer Purchase” box.

**Creating a Random Number Function for Customer Pay**

Now, design your algorithm to get a value for customer pay. The value should be randomly generated based on the purchase amount. You may create your own logic/conditional statements for the value.

For example, if the purchase amount is £21.55, you may want customer to pay £25.00 (create your own rules!).

In order to remove the decimal point, ***ceil*** function might be useful.

int randomPay = ceil(randomPurchase); // from this code, you get £22.

Now you need if else statements to get £25.00 such as,

if (randomPay % 10 == 2) //when the remainder is 2

{

randomPay += 3; // to get £x5.00

}

else if (randomPay % 10 == 5)

{

randomPay += 1; // to get £x6.00

}

**. . . more codes . . .**

Start writing your function for double DrawingTool::getRandomPay(). Write this function in ***DrawingTool.cpp***. Don’t forget to declare the prototype in ***DrawingTool.h***.

**Create a Clickable Image**

Making an object that gives an action to clicks is the same like *mousemove* we learned in lesson 1. See the code below and implement yours.

void DrawingTool::onLButtonDown(UINT nFlags, int x, int y)

{

cx = x;

cy = y;

if (cx > 650 && cx < 720 && cy >170 && cy < 230) //if the clear area is clicked

{

denom[0] = 0; // Declare “denom” variable in as a global

variable in DrawingTool.h

}

onDraw(); //you must always call your canvas function to take effect on drawing.

}

**Reminder:** Don’t forget to declare ***cx*** and ***cy*** in ***DrawingTool.h***!

In this example, we want to increase the count of “One Penny” when the box is clicked, include this code in void DrawingTool::onLButtonDown.

if (cx > 50 && cx < 150 && cy >300 && cy < 380)//if the one penny image is clicked

{

denom[0] += 1;

Why it increments by 1 for index 0?

}

This “denom” variable should be declared in DrawingTool.h file as a global variable.

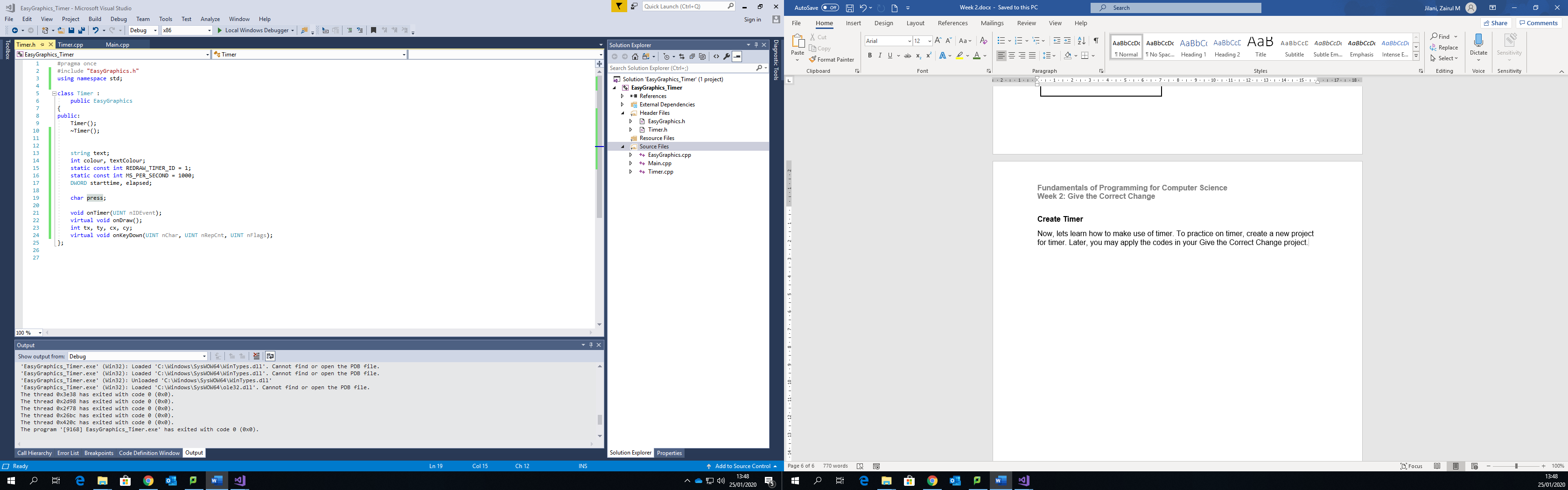
int denom [10] = {0,0,0,0,0,0,0,0,0,0};

Why the size of the array is 10?

**Create Timer**

Now, let’s learn how to make use of timer. To practice on timer, create a new project for timer to avoid mess in your current project. Later, you may apply the codes in Give the Correct Change.

Assuming you have these files (follow the steps in lesson 1).



In ***Timer.h***, declare the following variables and functions.

protected:

static const int REDRAW\_TIMER\_ID = 1;

static const int MS\_PER\_SECOND = 1000;

DWORD starttime, elapsed;

protected:

void onTimer(UINT nIDEvent);

virtual void onDraw(); //your canvas for drawing!

int tx, ty, cx, cy; //t:text, c:circle

virtual void onKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);

//additional variables for this example

string text;

int colour, textColour;

Don’t forget to include using namespace std; in your files!

In ***Timer.cpp*** write the following codes.

Set these global variables values (you may set your preferred values)

colour = GREY;

textColour = BLUE;

tx = 150;

ty = 200;

cx = 400;

cy = 350;

text = "HELLO";

setImmediateDrawMode(false); //this is fixed!

Then write the onDraw function

void Timer::onDraw()

{

clearScreen(colour);

text.c\_str() is where you set the value for the text variable

setBackColour(GREEN);

setPenColour(BLACK, 3);

setFont(30, L"Tahoma");

setTextColour(textColour);

drawText(text.c\_str(), tx, ty);

drawCircle(cx, cy, 35, true);

Discuss with your friends about this line of codes

elapsed = ::GetTickCount() - starttime;

int sec = elapsed / MS\_PER\_SECOND;

if (sec > 3)

{

textColour = YELLOW;

}

if (sec == 6 || sec == 8 || sec == 10)

Use mathematical equations to get interesting outputs

{

cx += 10\*sin(cx);

cy += 10\*sin(cy);

}

EasyGraphics::onDraw(); **//don’t forget to end the onDraw function with this code!**

}

Then write ***onTimer***

void Timer::onTimer(UINT nIDEvent)

{

if (nIDEvent == REDRAW\_TIMER\_ID)

onDraw();

}

Write this function for onKeyDown

void Timer::onKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

switch (nChar)

{

case VK\_RETURN:

starttime = ::GetTickCount(); //start the timer

setTimer(REDRAW\_TIMER\_ID, 60);

ty += 5;

textColour = BLACK;

text = "Start";

break;

case VK\_LEFT:

tx -= 5;

killTimer(REDRAW\_TIMER\_ID);

textColour = RED;

text = "Timer has been stopped!";

break;

}

onDraw(); // always call the onDraw function.

}

See the output and discuss with your instructor and friends, how can you apply timer in Give the Correct Change?.