

Scrum Update III (YANNIX)

☰ Tags	Scrum Update
🕒 Create	@March 13, 2022 5:37 PM
🔍 Nickname	
🕒 Last edited time	@March 22, 2022 2:31 PM
🔍 Site	
➦ Student	
🔗 URL	
☰ Scrum Update	[Last week] - ศึกษาชนิดของ Curve ที่ใช้ใน Computer design และเขียนโค้ดสำหรับการวาด Curve - เข้าไปศึกษา YANNIX Coding style กับพี่ที่บริษัท -ศึกษาการใช้งาน FLTK ขึ้นต้น [This week] -ศึกษา Model view projection metrix ของ OpenGL เนื่องจากอาทิตย์ที่แล้วมีหัวข้อเรื่อง Curve ที่ Supervisor ต้องการให้เรียนก่อน -Basic Image processing [Blocks] -การบ้านที่สั่งในอาทิตย์นี้
➦ Week	
➦ Scores	
📌 Status	

Resource:

Bezier curve1: <https://youtu.be/aVwxzDHniEw>

<https://youtu.be/aVwxzDHniEw>

Bezier curve2: <https://youtu.be/pnYccz1Ha34>

<https://youtu.be/pnYccz1Ha34>

B-Spline: <https://youtu.be/ghQrRCJ-mVg>

<https://youtu.be/ghQrRCJ-mVg>


FLTK: https://pyfltk.sourceforge.io/docs/CH0_Preface.html

My Note:


Code source: <https://github.com/use555555/Onboard/tree/main/Basic>

Onboard/Basic at main · use555555/Onboard

Contribute to use555555/Onboard development by creating an account on GitHub.

 <https://github.com/use555555/Onboard/tree/main/Basic>

use555555/
Onboard



1 Contributor 0 Issues 0 Stars 0 Forks

Code ของครั้งนี้จะอยู่ในส่วนของ FLTK และ OpenGL ที่อยู่ในส่วนของ Bezier, B-spline, Circle

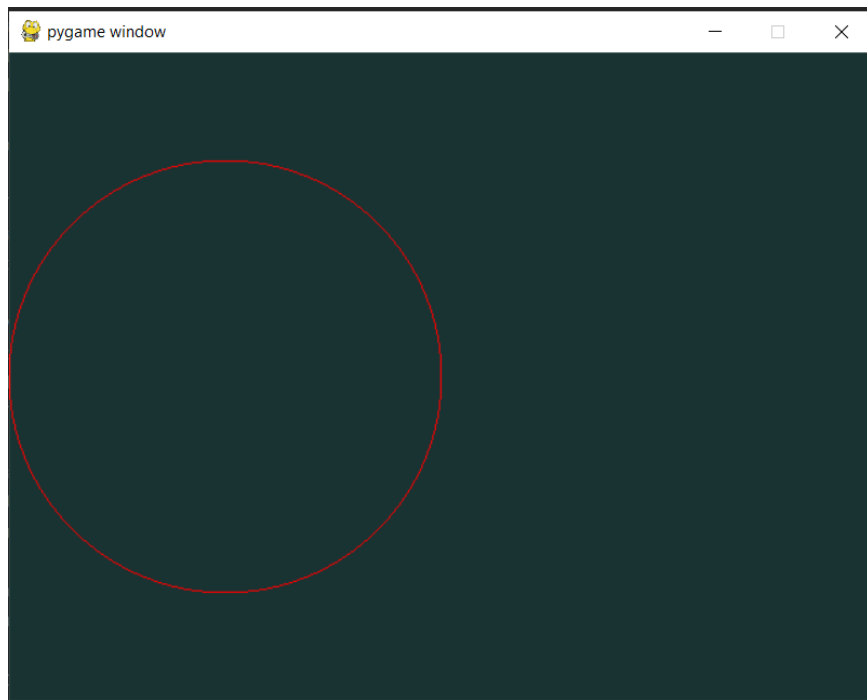
Curve

Circle

ในส่วนของการสร้างวงกลมส่วนนี้ผมทำด้วยความเข้าใจของสมการวงกลมและพิกัดเชิงมุมในการสร้างรูปวงกลมซึ่ง Input ของ Code ผมจะเป็นการใส่ พิกัด x, y ของสองจุด ซึ่งจุดแรกจะเป็นจุดศูนย์กลางและจุดที่สองจะมีไว้ ทำการคำนวณหาค่า พอดีรัศมีแล้วจะนำมาหาพิกัดบนเส้นรอบวงด้วยสมการ

$$x = r\cos(t) + cx \quad y = r\sin(t) + cy$$

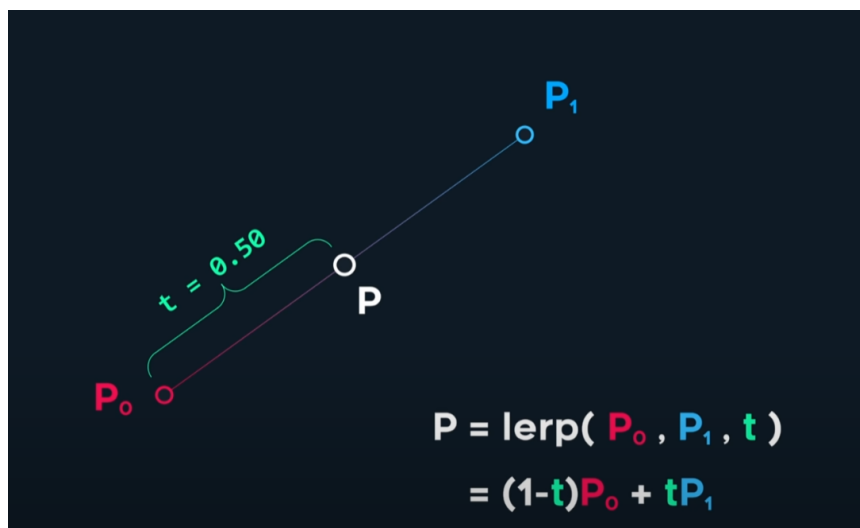
ซึ่งจะได้พิกัด x, y ใน Step ตามที่เราได้กำหนดไว้ ซึ่งแต่ละ step จะนำมาใส่เป็นค่าของ t ในสมการตัวอย่าง ใน Code ที่พิกัดที่ใส่ไปเป็น (-0.5, 0) และ (0, 0) ซึ่งจะเป็นพิกัดที่ใช้ถูก Normalize เพื่อใช้กับ OpenGL จะได้ Output ดังนี้



Bezier

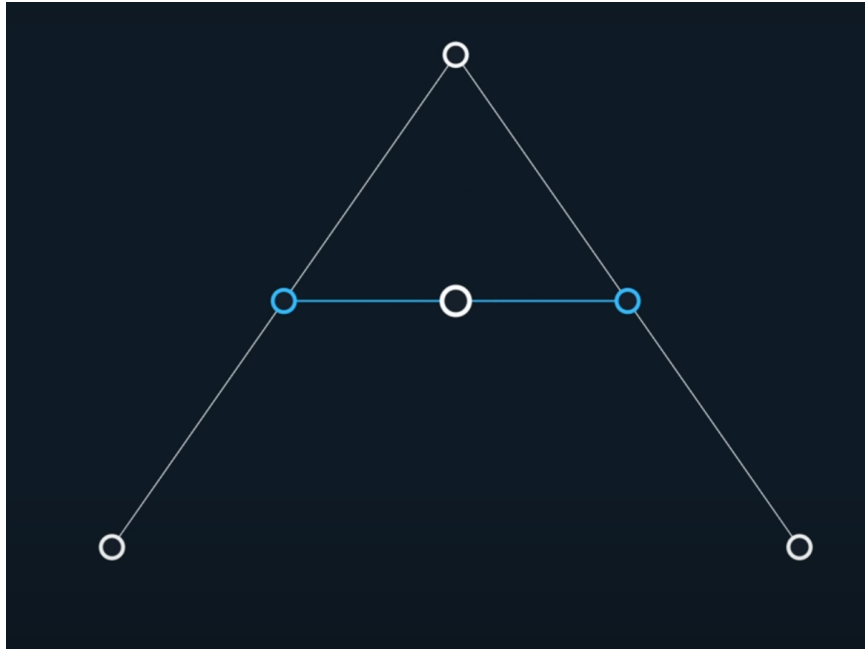
ความเข้าใจคร่าวๆ ของ Bezier คือการที่ทำการ Interpolate ระหว่างจุดที่ต่อกันซึ่งในการ Interpolate แต่ละครั้งจะส่งผลให้สมการสุดท้ายมี Degree ที่สูงขึ้น ตามจำนวนครั้งที่ Interpolate โดยจำนวนครั้งที่ Interpolate จะมาจากจำนวนของ จุดที่จะมากำกับ เช่นถ้าใช้ใช้ จุด 2 จุดกำกับเส้น จะทำการ Interpolate 1 ครั้ง ก็จะได้เป็นสมการ Linear ซึ่งจะมีรูปแบบดังนี้

Linear

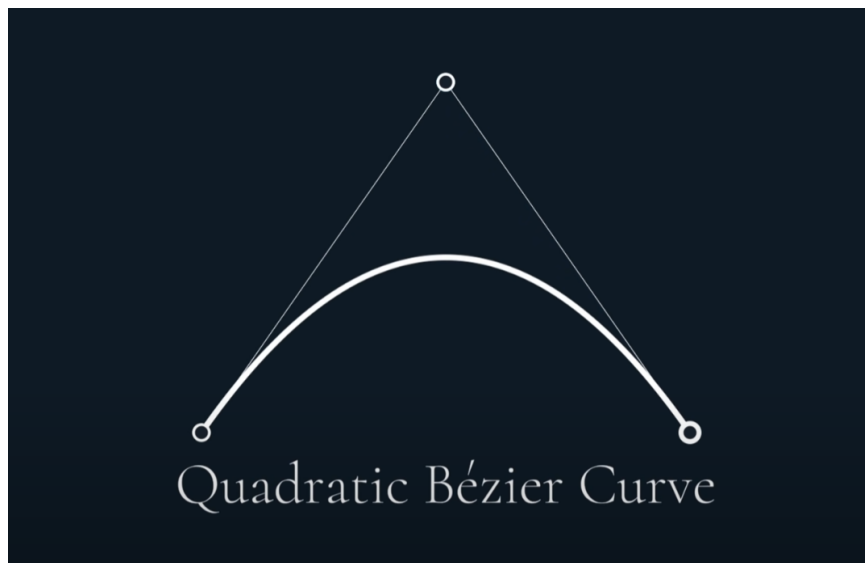


ถ้าใช้ใช้จุด 3 จุดกำกับเส้น จะทำการ Interpolate 2 ครั้ง ก็จะได้เป็นสมการ Quadratic ซึ่งจะมีรูปแบบดังนี้

Quadratic



จะทำให้ได้รูป



ซึ่งจะมีสมการ ดังนี้

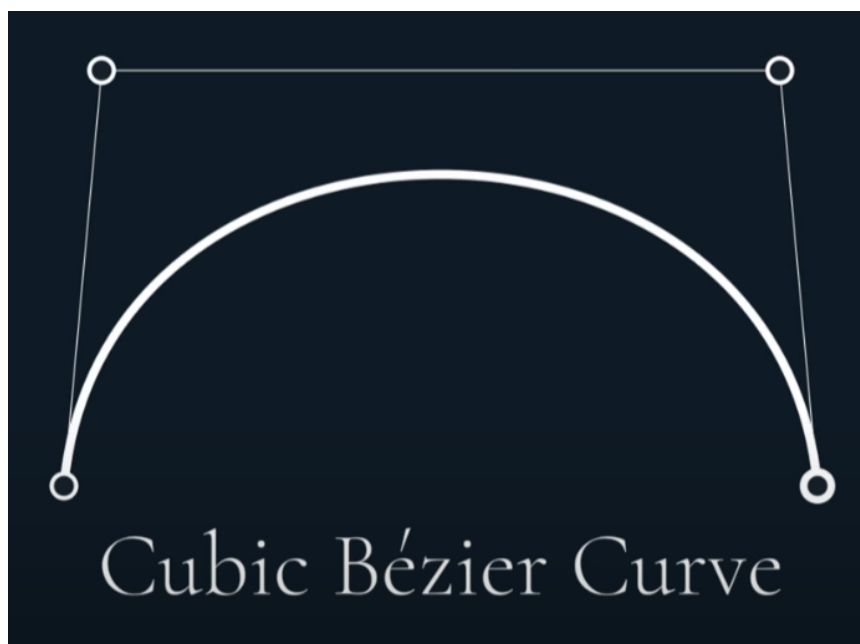
$$\begin{aligned}
 L_1(t) &= (1 - t)P_1 + tP_2 \\
 Q_0(t) &= (1 - t)L_0(t) + tL_1(t) \\
 \underline{Q_0(t) &= (1 - t)^2P_0 + 2(1 - t)tP_1 + t^2P_2}
 \end{aligned}$$

ถ้าใช้จุด 3 จุดกำกับเส้น จะทำการ Interpolate 3 ครั้ง ก็จะได้เป็นสมการ Cubic ซึ่งจะมีรูปแบบ ดังนี้

Cubic



จะทำให้ได้รูป

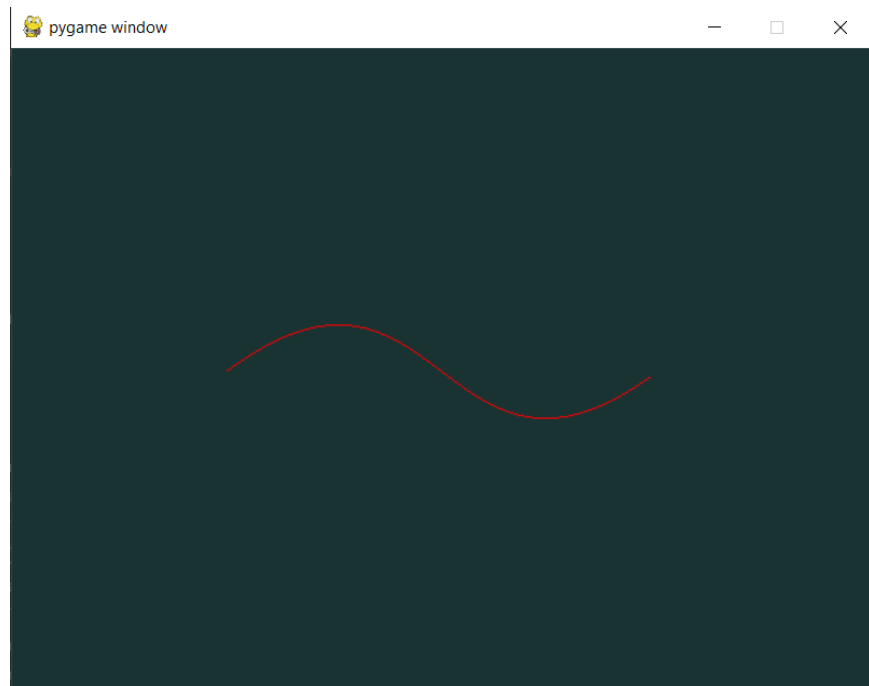


ซึ่งจะมีสมการ ดังนี้

$$\begin{aligned}L_2(t) &= (1 - t)P_2 + tP_3 \\Q_1(t) &= (1 - t)L_1(t) + tL_2(t) \\C_0(t) &= (1 - t)Q_0(t) + tQ_1(t) \\C_0(t) &= (1 - t)^3P_0 + 3(1 - t)^2tP_1 + 3(1 - t)t^2P_2 + t^3P_3\end{aligned}$$

จากข้างต้นจะเป็น Bezier curve ที่มีการใช้งานบ่อย แต่เราก็สามารถใช้จำนวน จุดกำกับที่มากขึ้นได้แต่ก็จะส่งผลให้สมการมี degree ที่สูงขึ้น

จากที่ผมได้ทำความเข้าใจผมได้ไปทำการเขียน Code เพื่อทำการ Generate Bezier curve ที่สามารถกำหนดจุดได้กี่จุดก็ได้ ซึ่งตัวอย่างด้านล่างนี้ จะเป็นตัวอย่างที่ผมกำหนดจุดไป 4 จุดซึ่งมี $(-0.5, 0.0)$, $(0.0, 0.5)$, $(0.0, -0.5)$ และ $(0.5, 0.0)$ ตามลำดับ จะได้ Output



แต่ใน Video แรกที่ผมไปดูนั้น มีข้อมูลที่มากกว่านี้แต่จะเป็นส่วนที่ผมไม่ได้นำไปใช้ในงานนี้ผมจะขอไม่ทำการสรุปนะครับ

B-Spline

ในการทำ B-Spline curve จะเป็นการนำ Bezier curve มาต่อกัน ซึ่งจำนวน Curve จะถูกกำหนดโดยจุดกำกับและ Degree ของ curve เช่น

จะสร้าง Cubic b-spline curve (degree = 3) และมี 6 control points จะทำให้มีจำนวน Curve ที่จะถูก output ออกมาจะมีจำนวน $6 - 3 = 3$ curve

โดย B-spline นั้นก็จะมี property อยู่คือ

1. จุดสุดท้ายของ Bezier curve แรกต้องต่อกับจุดต้นของ Bezier curve ของเส้นต่อไป (C0 continuity)
2. First derivative ที่จุดสุดท้ายของ curve แรกต้องมีค่าเท่ากับจุดเริ่มต้นของ curve ต่อมา (C1 continuity)

3. Second derivative ที่จุดสุดท้ายของ curve แรกต้องมีค่าเท่ากับจุดเริ่มต้นของ curve ต่อมา (C2 continuity)

โดยสมการของ B-spline จะเป็นดังรูป

$$S(t) = \sum_{i=0}^n N_{i,k}(t) P_i,$$

ซึ่ง k คือ degree ของ curve, n คือจำนวน control point - 1, N คือ Basis function

ซึ่ง Basis function จะมีการคำนวณ โดย Cox - de Boor recursion formula ซึ่งจะมีสมการดังภาพ

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases},$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t).$$

ถ้าดู $N_{i,k}$ ของสมการ b-spline ถ้ามี control point 3 จุด จะทำให้ต้องหา $N_{0,k}, N_{1,k}, N_{2,k}$ เพื่อมา sum ในสมการ b-spline ซึ่งจากรูป ของสมการ Cox - de Boor recursion เราสามารถกำหนด $N_{0,0}, N_{1,0}, N_{2,0}, \dots$ ที่เวลา t ใดๆ ได้แล้วจะต้องใช้สมการที่ 2 ในการคำนวณ recursion ให้ได้ถึง Degree ที่กำหนดซึ่งจะทำให้การคำนวณเป็นสามเหลี่ยมดังภาพ

$$\begin{array}{ccccccc} N_{0,0}(t) & \rightarrow & N_{0,1}(t) & \cdots & N_{0,k-1}(t) & \rightarrow & N_{0,k}(t) \\ & \nearrow & & & & \nearrow & \\ N_{1,0}(t) & \rightarrow & N_{1,1}(t) & \cdots & N_{1,k-1}(t) & & \\ & \vdots & & & \vdots & & \\ N_{n-1,0}(t) & \rightarrow & N_{n-1,1}(t) & & & & \\ & \nearrow & & & & & \\ N_{n,0}(t) & & & & & & \end{array}$$

เมื่อได้ Basis function ที่ต้องใช้แล้วก็จะนำไปคูณกับ control point จุดที่ i ตามสมการของ b-spline แล้วเราก็ทำการ sum ทั้งหมดที่ได้คูณมา ซึ่งก็จะได้พิกัด ณ เวลา นั้น

ในสมการของ Cox - de Boor recursion จะมีอีกส่วนที่มีสัญลักษณ์ t_i ซึ่งจะเป็นค่า knot ที่อยู่ใน knot vector ที่จะส่งผลกับรูปร่างของ curve ที่สมการจะถูกคำนวณออกมาซึ่ง knot vector จะมีหน้าตาดังนี้

$$\mathbf{T} = (t_0, t_1, \dots, t_m).$$

ซึ่งจำนวนของ knot จะคำนวณจาก degree + จำนวน control point + 1

การกำหนด knot vector จะทำให้เกิด curve แบบต่างๆ แต่ในส่วนนี้ผมยังไม่ได้ลงข้อมูลสักผมจะขอจบสรุปไว้เท่านี้ก่อน แต่ก็ผมได้ไปทำ code ในการวาด curve นี้ไปแล้วแต่ซึ่งผมได้กำหนด degree เป็น 2, control point คือ (-0.5, 0.0), (0.0, 0.5), (0.0, -0.5) และ (0.5, 0.0) และผมกำหนดให้เป็น open curve ซึ่งจะเป็นการกำหนด knot vector ที่มีการซ้ำตัวหน้าและตัวหลัง (ในส่วนนี้ผมขอออกคร่าวๆ ไว้เท่านี้) ซึ่งจะได้ output ดังภาพ

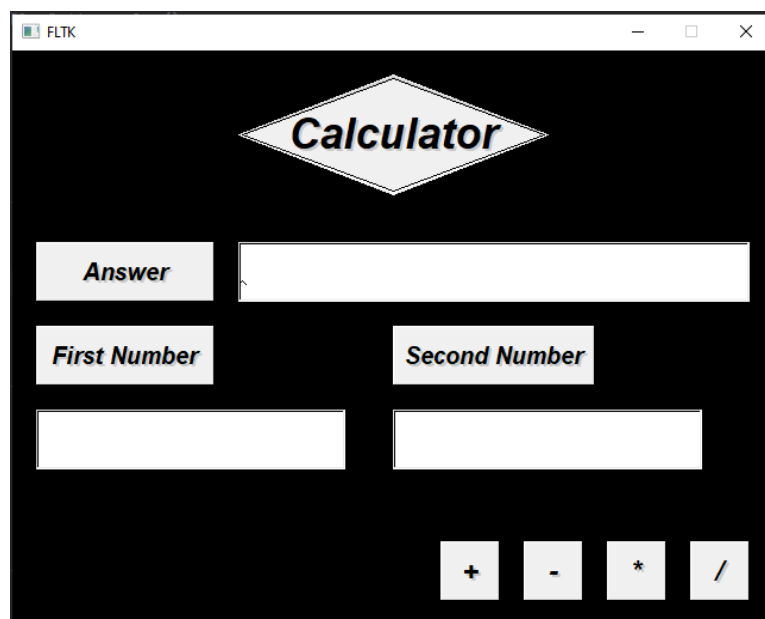


FLTK

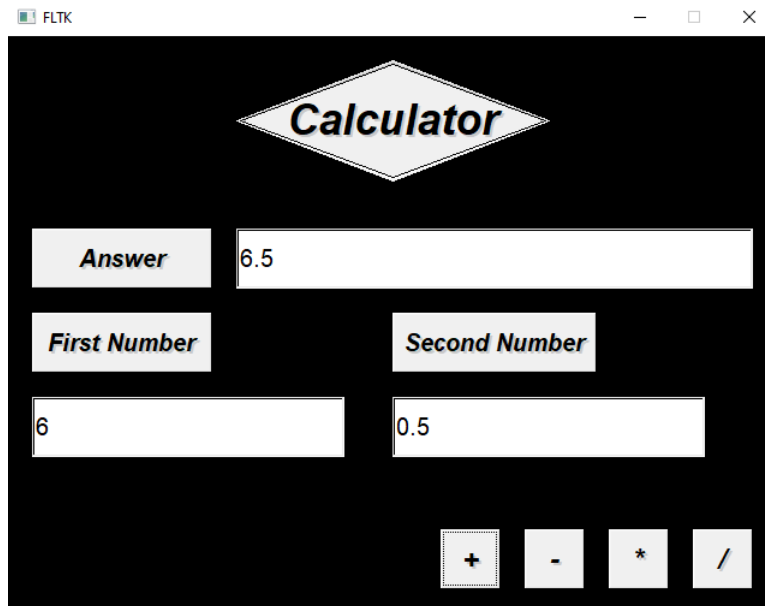
ในส่วนนี้ผมได้ดูไปคร่าวๆ ซึ่งที่ผมไปดูจะเป็นเรื่องของการเปิด window, การสร้าง Text, การสร้างปุ่ม และการสร้างช่องสำหรับ Input-output บน UI ซึ่งผมได้ทำการเขียนเล่นๆ เป็น file hello world แรกที่ สามารถแสดง text มีปุ่มและ มีช่องสำหรับ input-output string ซึ่งจะมีดังนี้



แต่ผมก็ได้นำสิ่งที่ผมเขียนเล่นๆ นี้ไปสร้างโปรแกรม Calculator แบบ Basic ดังภาพ



ซึ่งสามารถคำนวณเลขทศนิยมหรือจำนวนเต็มสองเลขมาทำการบวก, ลบ, คูณ, หารได้ซึ่งจะเป็นตัวอย่างด้านล่างที่จะเป็นการบวก



YANNIX coding style

ในส่วนนี้ผมได้ไปศึกษาที่บริษัทซึ่งจะเป็นรูปแบบการพิมพ์ Code ที่ใช้ในการทำงานในบริษัทซึ่งจะเป็นรูปแบบการเขียนที่จะทำให้คนในบริษัทที่ไม่ได้เขียน Code สามารถเข้าใจการทำงานของ Code ที่เขียนได้ ซึ่งผมสามารถพูดได้ดังนี้

1. การเขียน ตัวแปรใน Code แบบ Camel case ซึ่งจะมีวิธีการเขียนที่ต่างตามการใช้งาน
2. การเขียน Comment อธิบายการทำงานของ Class
3. การเขียน ตัวแปรที่ใส่ในวงเล็บที่ต้องทำการเว้น ตรงจุดที่จะต่อกับวงเล็บ เช่น (x)