

國立虎尾科技大學

機械設計工程系

電腦輔助設計實習 ag4 期末報告

3D 列印機

3D Printer

學生：

設計二甲 40623109 李如芳

設計二甲 40623124 葉修宏

設計二甲 40623127 張育偉

設計二甲 40623128 張華倞

設計二甲 40623130 陳鉅忠

設計二甲 40623154 黃馨慧

指導教授：嚴家銘

20190110

摘要

列印機外觀

列印機使用材料

V-rep 模擬的設定過程

V-rep python remote api 編寫過程

Onshape 自訂義功能練習過程

目錄

摘要	i
目錄	ii
表目錄	iii
圖目錄	iv
第一章 前言	1
第二章 列印機外觀	2
第三章 列印機使用材料	4
第四章 V-rep 模擬	5
4.1 First-step	5
4.2 Second-step	6
4.3 Third-step	8
4.4 Final-step	9
4.5 test	10
第五章 V-rep python remote api	12
第六章 Onshape 自訂義功能	15
6.1 如何使用自訂義的功能	15
6.2 如何建立新的 Feature studio	15
6.3 建立表單介紹	17
6.4 自製繪圖功能練習	23
第七章 參考文獻	24

表目錄

表 3.1 使用材料表	4
-----------------------	---

圖目錄

圖 2.1 列印機外觀	3
圖 5.1 Function xyz-position	12
圖 5.2 Function get-xyz-position	12
圖 5.3 Function xyz-position-2	13
圖 5.4 Function xyz-move	13
圖 5.5 Function move-back	14
圖 6.1 自訂義功能使用位置	15
圖 6.2 新的 Feature studio 開啟位置	16
圖 6.3 快捷指令列	17
圖 6.4 新的特徵	18
圖 6.5 特徵格式	19
圖 6.6 特徵編輯內容	20
圖 6.7 插入表單	21
圖 6.8 Length	22
圖 6.9 長度表單	22
圖 6.10 星星練習成果	23

第一章 前言

3D 列印機對於一個設計者而言是一樣很重要工具，可以大量縮短製程時間，但往往買一台的價格太過昂貴，因此我們決定製作一台自組 3D 列印機，大小可以依照個人需求更改，且自行組裝過程可以學習到許多東西，其中包括機構設計、電路分析、程式設計等等。

我們最終目標為人人皆可以自行組裝一台低成本的 3D 列印機。

第二章 列印機外觀

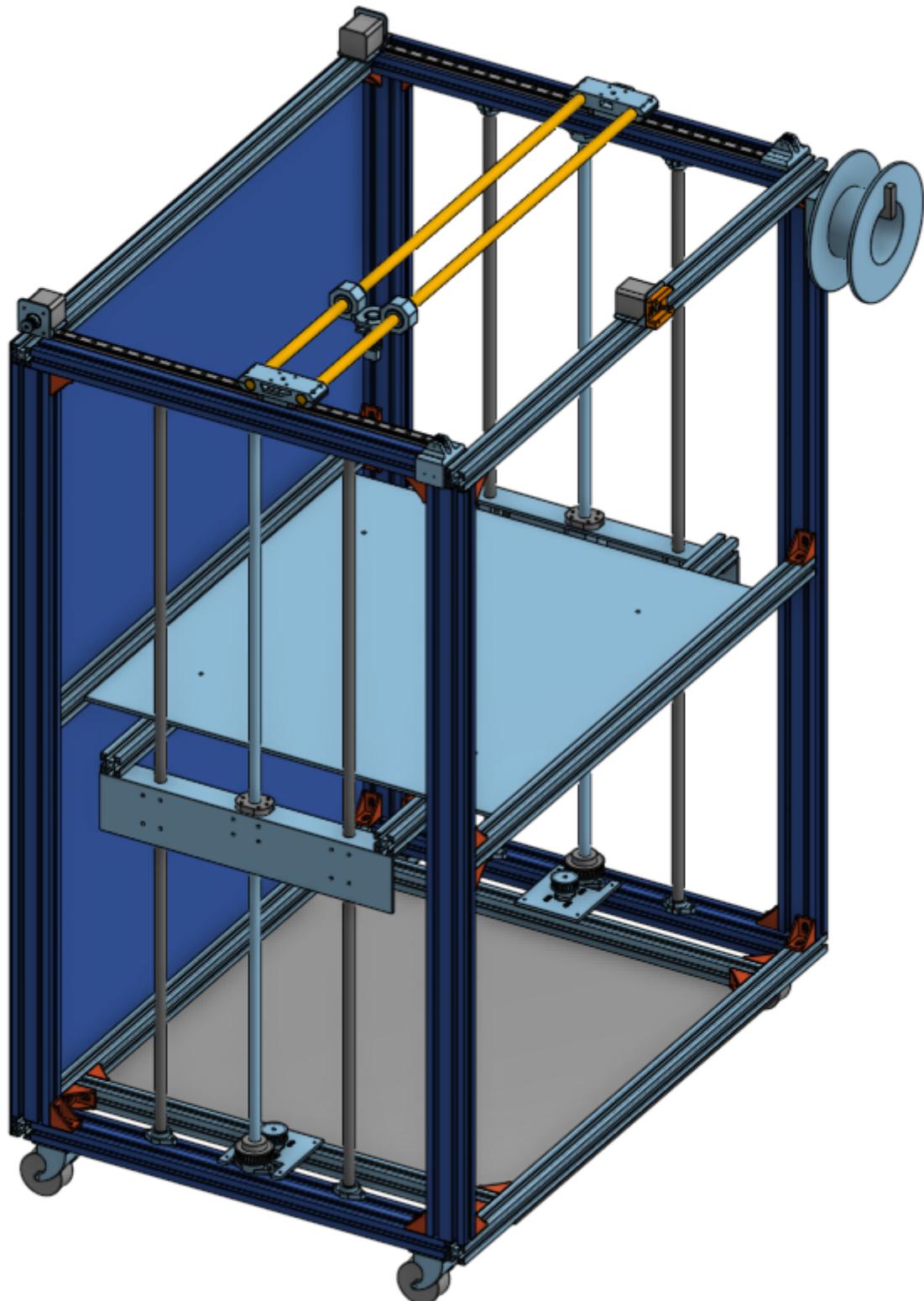


圖 2.1: 列印機外觀

第三章 列印機使用材料

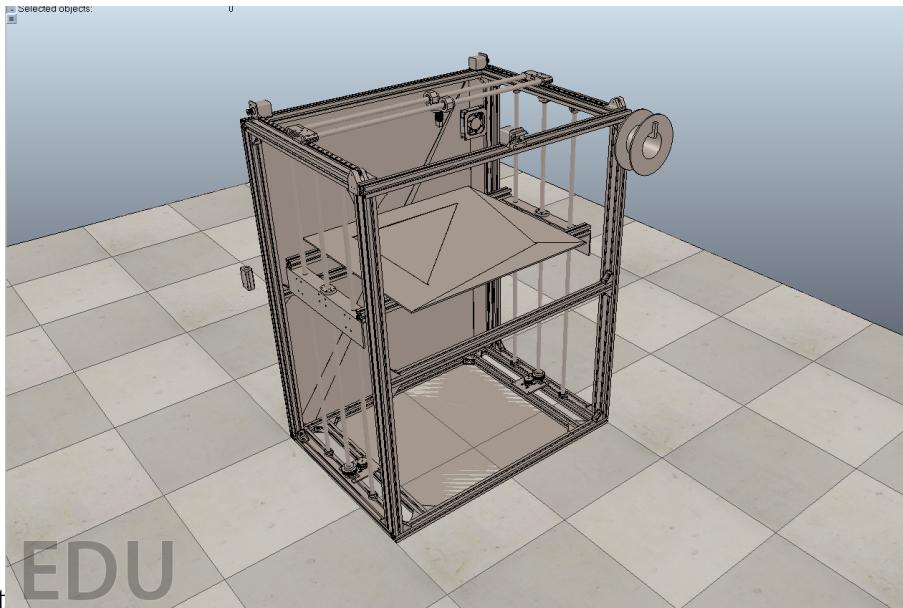
表 3.1: 使用材料表

材料名稱	超連結
腳輪	連結
螺母座	連結
42 部馬達支架	連結
光軸滑塊	連結
噴頭	連結 1 連結 2
噴嘴	連結
風扇	連結
同步輪	連結 1 連結 2
軸座	連結
線性滑塊	連結 1 連結 2
光軸固定座	連結 1 連結 2
惰輪 2GT	連結

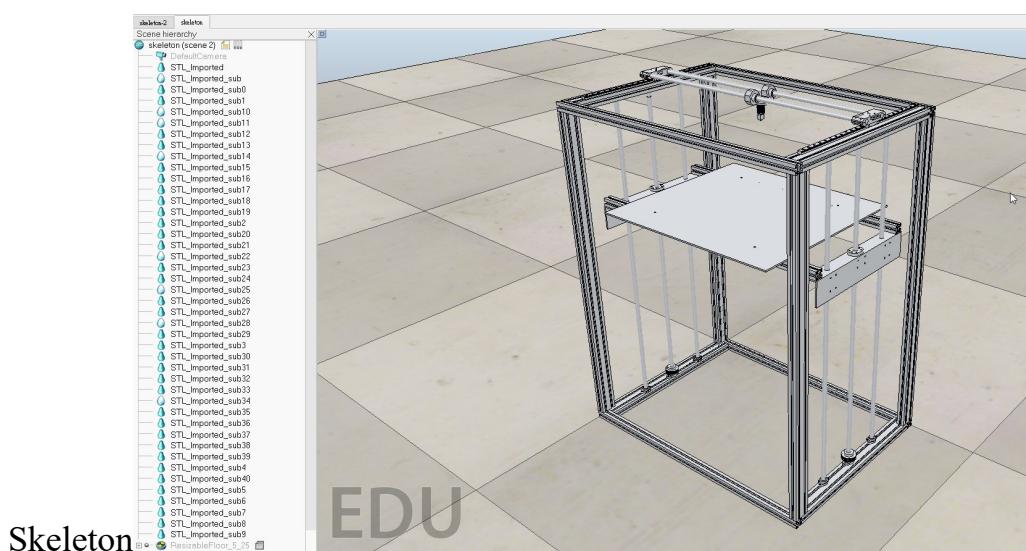
第四章 V-rep 模擬

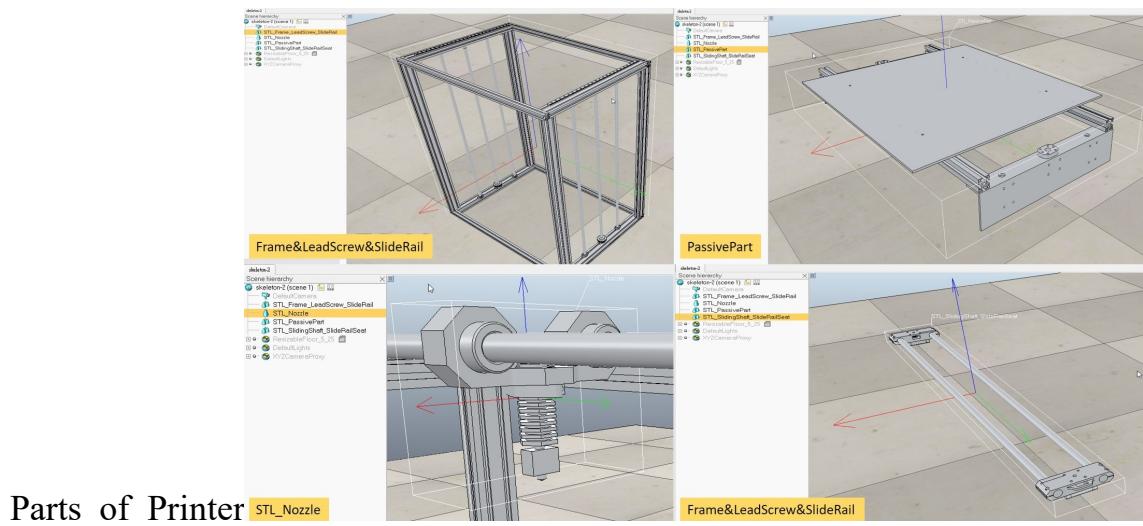
4.1 First-step

再模擬主機前先試著使用 V-rep 這套軟體 PDF_Hit_Me



大多都是馬達作動皮帶並帶動齒輪運動，考慮到系統運作的流暢度、物件的可視性及操作的方便性，我將原圖形“簡化”並分成四個部分，分別是：





Parts_of_Printer **STL_Nozzle**

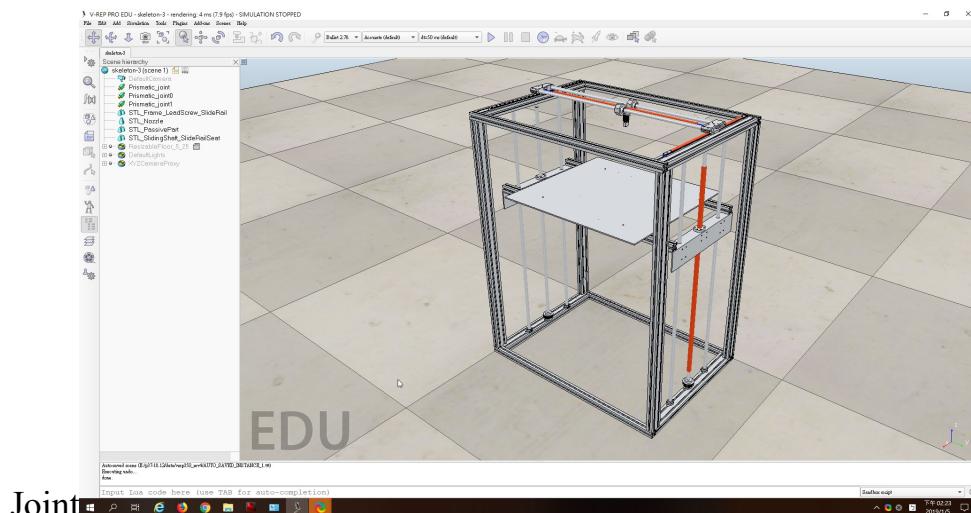
Frame&LeadScrew&SlideRail：影響最大的支架 & 導螺桿 & 軸 & 滑軌

Nozzle：最上方的噴嘴

SlidingShaft&SlideRailSeat：移動噴嘴的滑軌座 & 滑動軸

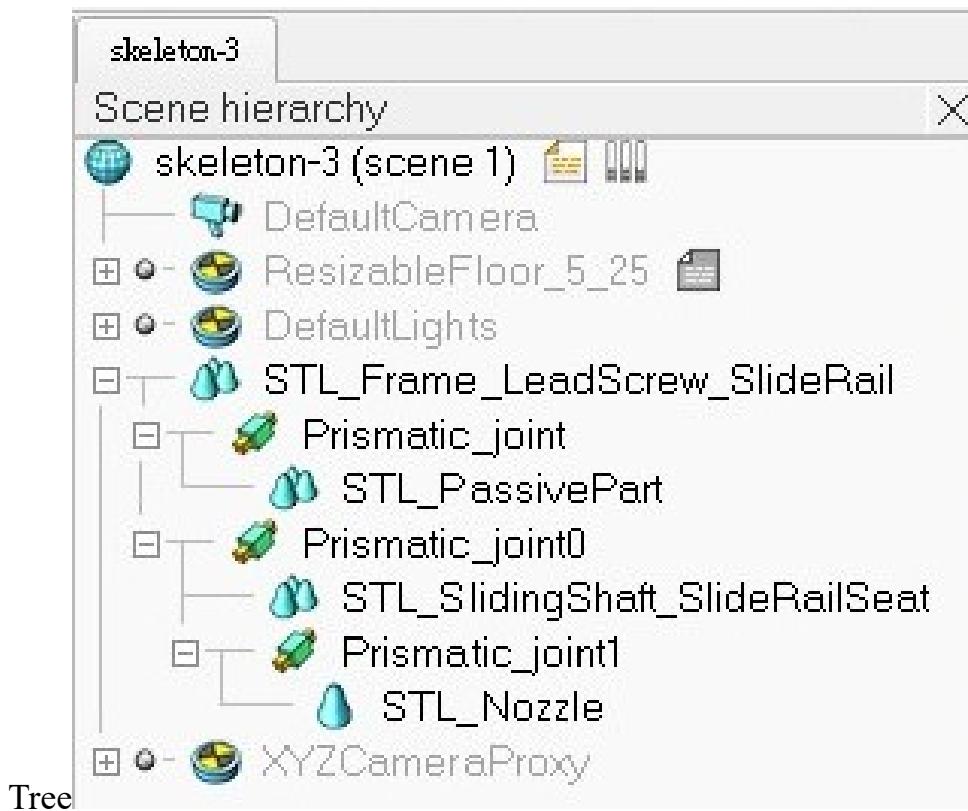
PassivePart：放置作品的平板 & 保持平衡的滾珠導螺桿座

4.2 Second-step



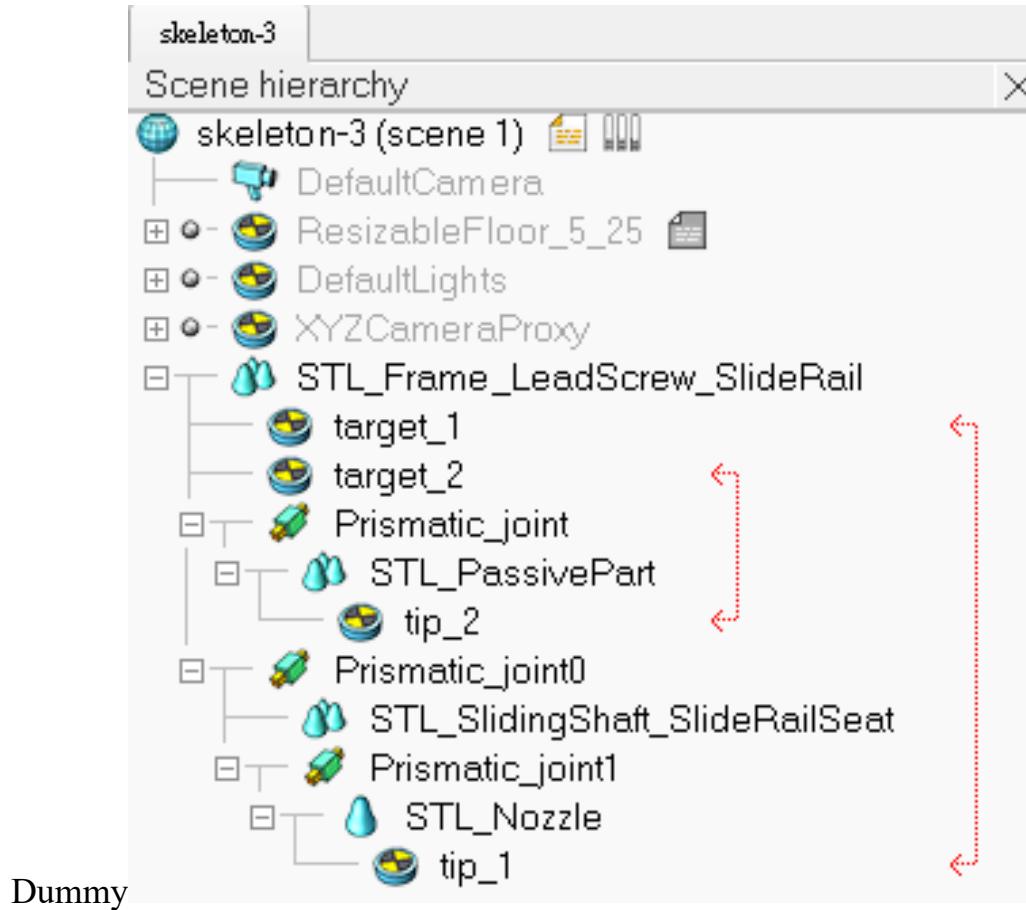
Joint

加入方形的 joint : prismatic，並將他們設定到作動的位置，我解化了各個軸的 joint
並設定成最重要的選擇三個 XYZ 的軸，並設定好運動範圍必免模擬時超出範圍

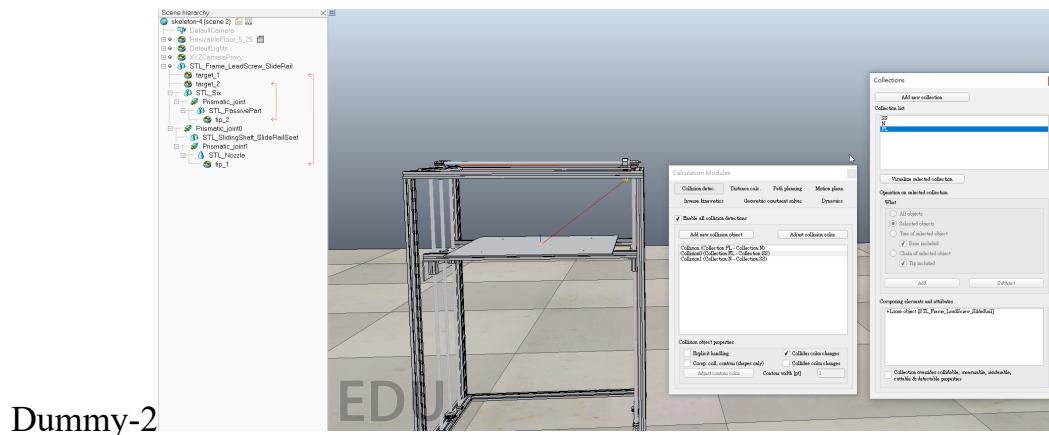


設定好樹狀圖關係

4.3 Third-step



加入四個 Dummy 後設定好運動關係，我選擇使用 IK 運動算試於 tip1→target1 與 tip2→target2



在設定 collision 時發現 Frame&LeadScrew&SlideRail 這物件包含的 Lead-Screw&SlideRail 滑動軸與導螺感會與 PassivePart 中間的滾珠導螺桿座及平板物

件會直接發生干涉，所以我將它們分開並命名為 Six

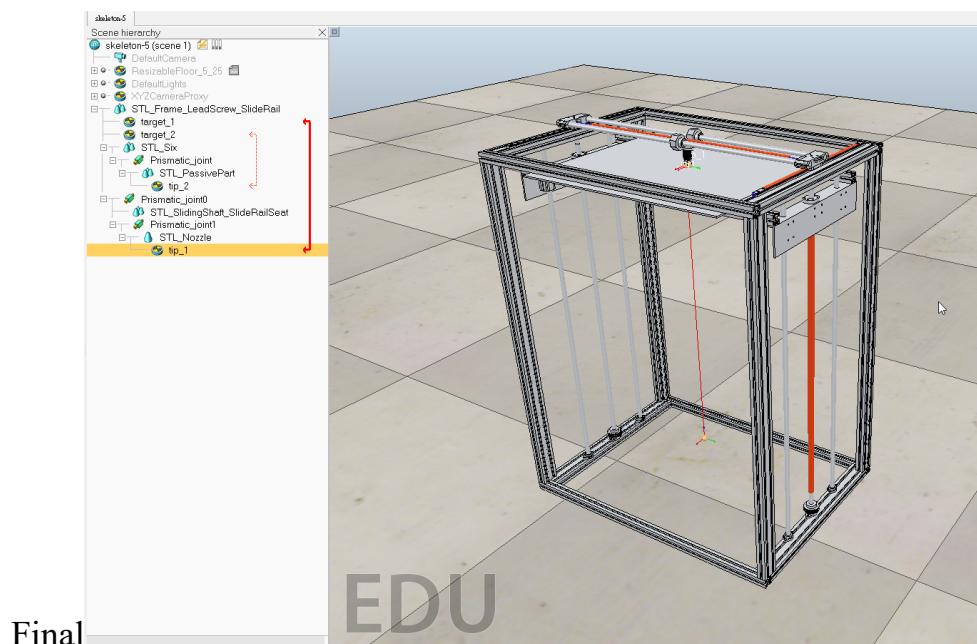
設定 collision:

N(中上方的噴頭) 與 FL(Printer 骨架)

N(中間的噴頭) 與 SS(Printer 骨架)

SS(中間的滾珠導螺桿座及平板物件) 與 FL(Printer 骨架)

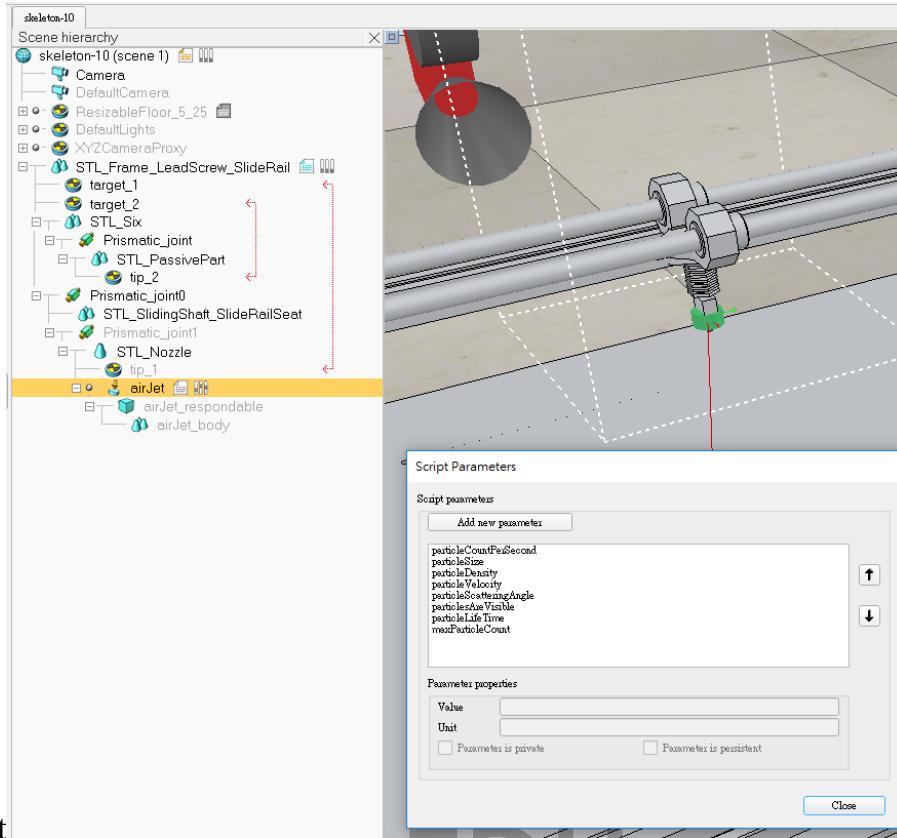
4.4 Final-step



除了 code 的部分，V-rep 設定的大概就這樣了。

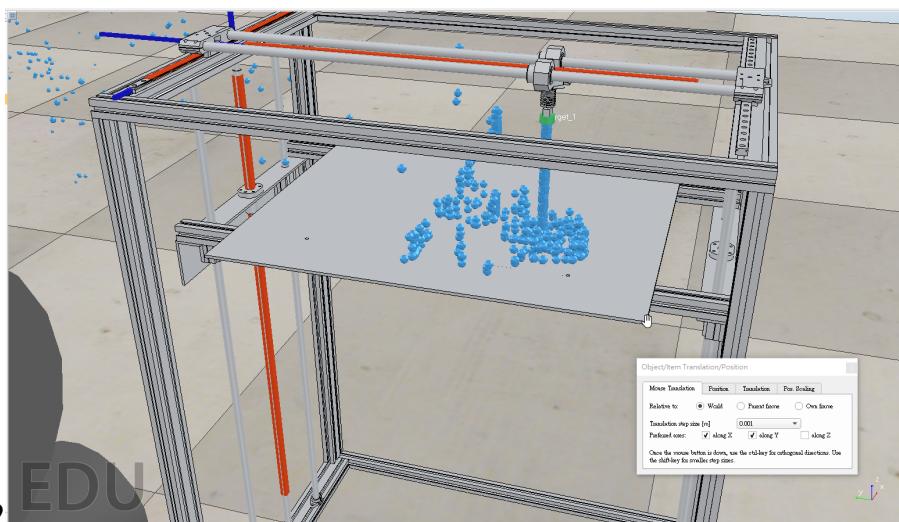
基本滑動模擬影片: https://www.youtube.com/watch?v=V_GmofG5xhE

4.5 test



test

嘗試添加噴頭並測試噴射填料，利用檔案中的 air jet 模擬，設定好位置後於程式碼中調整模擬噴料口徑及利用 Script Parameter 設定噴料粒子大小、初速、數量、存留時間等等後，將 PassivePart 設為運動反應物件。



test2

就能測試是否能作動，目前卡在程式碼編寫粒子堆疊的編寫，嘗試將 simCreateStack 寫入並設定。

Home

Regular API function

simCreateStack	
Description	Creates a stack object that is mainly used to exchange complex data in an easy way with scripts. See also simReleaseStack and the other stack functions .
C synopsis	simInt simCreateStack()
C parameters	-
C return value	-1 in case of an error, otherwise a stack handle.
Lua synopsis	-
Lua parameters	-
Lua return values	-

[All regular API functions on one page](#)

particlestack

第五章 V-rep python remote api

首先開始製作之前要先下載 v-rep 的 python 函式庫

下載連結

再來只要在程式的一開始匯入程式庫即可使用相關函式

各函式功能連結

程式架構

首先定義三個函式，用來分別定義 xyz 的位置如圖 5-1。

```
def setJointPositionz(poi, steps):
    for i in range(steps):
        errorCode=vrep.simxSetJointPosition(clientID, Prismatic_joint_handle, 0.11-i*poi, vrep.simx_opmode_oneshot_wait)

def setJointPositiony(poi, steps):
    for i in range(steps):
        errorCode=vrep.simxSetJointPosition(clientID, Prismatic_joint1_handle, i*poi-0.157, vrep.simx_opmode_oneshot_wait)

def setJointPositionx(poi, steps):
    for i in range(steps):
        errorCode=vrep.simxSetJointPosition(clientID, Prismatic_joint0_handle, i*poi-0.09, vrep.simx_opmode_oneshot_wait)
```

圖 5.1: Function xyz-position

但發現無法從當前位置繼續移動，會直接順移回至原點，因此需要增加每次移動後的座標位置，所以使用圖 5-2 的函式獲得當前座標，並加入圖 5-1 的函式變為如圖 5-3 的函式。

```
def getJointPosition_x():
    position_x=vrep.simxGetJointPosition(clientID,Prismatic_joint0_handle,vrep.simx_opmode_oneshot_wait)
    x = position_x[1]
    print(x)
def getJointPosition_y():
    position_y=vrep.simxGetJointPosition(clientID,Prismatic_joint1_handle,vrep.simx_opmode_oneshot_wait)
    y = position_y[1]
    print(y)
def getJointPosition_z():
    position_z=vrep.simxGetJointPosition(clientID,Prismatic_joint_handle,vrep.simx_opmode_oneshot_wait)
    z = position_z[1]
    print(z)
```

圖 5.2: Function get-xyz-position

```

def setJointPositionz(poi, steps):
    for i in range(steps):
        position_z=vrep.simxGetJointPosition(clientID,Prismatic_joint_handle,vrep.simx_opmode_oneshot_wait)
        z = position_z[1]
        print('z = ',z)
        errorCode=vrep.simxSetJointPosition(clientID, Prismatic_joint_handle, z-poi, vrep.simx_opmode_oneshot_wait)

def setJointPositiony(poi, steps):
    for i in range(steps):
        position_y=vrep.simxGetJointPosition(clientID,Prismatic_joint1_handle,vrep.simx_opmode_oneshot_wait)
        y = position_y[1]
        print('y = ',y)
        errorCode=vrep.simxSetJointPosition(clientID, Prismatic_joint1_handle, poi+y, vrep.simx_opmode_oneshot_wait)

def setJointPositionx(poi, steps):
    for i in range(steps):
        position_x=vrep.simxGetJointPosition(clientID,Prismatic_joint0_handle,vrep.simx_opmode_oneshot_wait)
        x = position_x[1]
        print('x = ',x)
        errorCode=vrep.simxSetJointPosition(clientID, Prismatic_joint0_handle, poi+x, vrep.simx_opmode_oneshot_wait)

```

圖 5.3: Function xyz-position-2

但每次移動都要更改輸入的函式太麻煩，所以在定義一個函式來藉由輸入座標進行移動如圖 5-4。

```

def goJointPosition(x,y,z):
    x_step = x*10
    y_step = y*10
    z_step = z*10
    setJointPositionx(0.001, x_step)
    setJointPositiony(0.001, y_step)
    setJointPositionz(0.001, z_step)

```

圖 5.4: Function xyz-move

但因為是利用 for 迴圈執行步數，單步行走 0.001m，所以無法倒退，因此加入 if 函式來判定輸入的值是否小於零。如圖 5-5，是判定 xyz>0 與 x<0、yz>0 的狀況。

設定檔與程式檔下載:[點我](#)

```
if x >= 0 and y >= 0 and z >= 0 :  
    x_step = x*10  
    y_step = y*10  
    z_step = z*10  
    setJointPositionx(0.001, x_step)  
    setJointPositiony(0.001, y_step)  
    setJointPositionz(0.001, z_step)  
elif x < 0 and y >= 0 and z >= 0 :  
    x_step = x*-10  
    y_step = y*10  
    z_step = z*10  
    setJointPositionx(-0.001, x_step)  
    setJointPositiony(0.001, y_step)  
    setJointPositionz(0.001, z_step)
```

圖 5.5: Function move-back

第六章 Onshape 自訂義功能

6.1 如何使用自訂義的功能

在零件圖功能區的最右端處點選即可使用。如圖 6.1

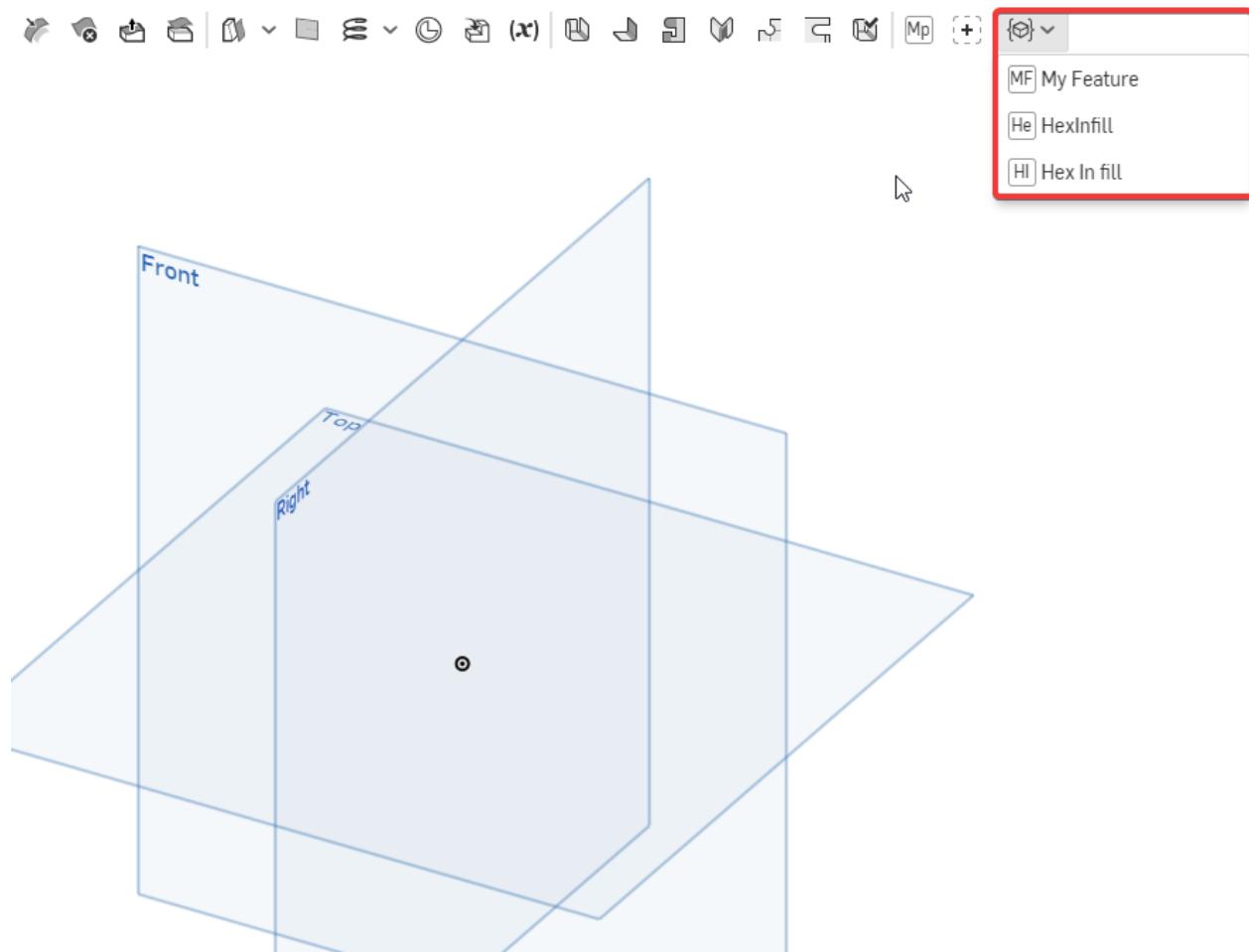


圖 6.1: 自訂義功能使用位置

6.2 如何建立新的 Feature studio

先開起任意一個圖檔，

於左下角加號處開啟。如圖 6.2

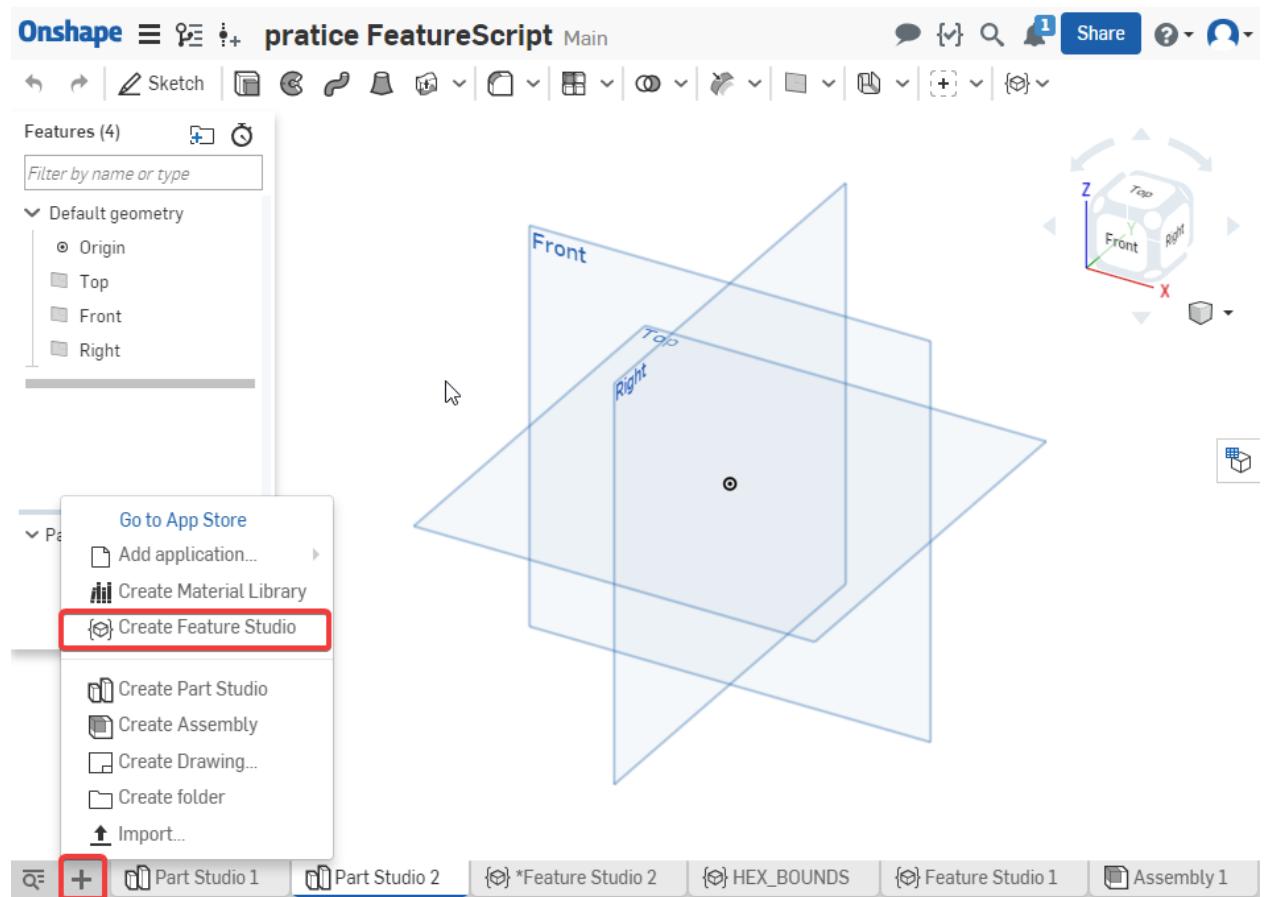
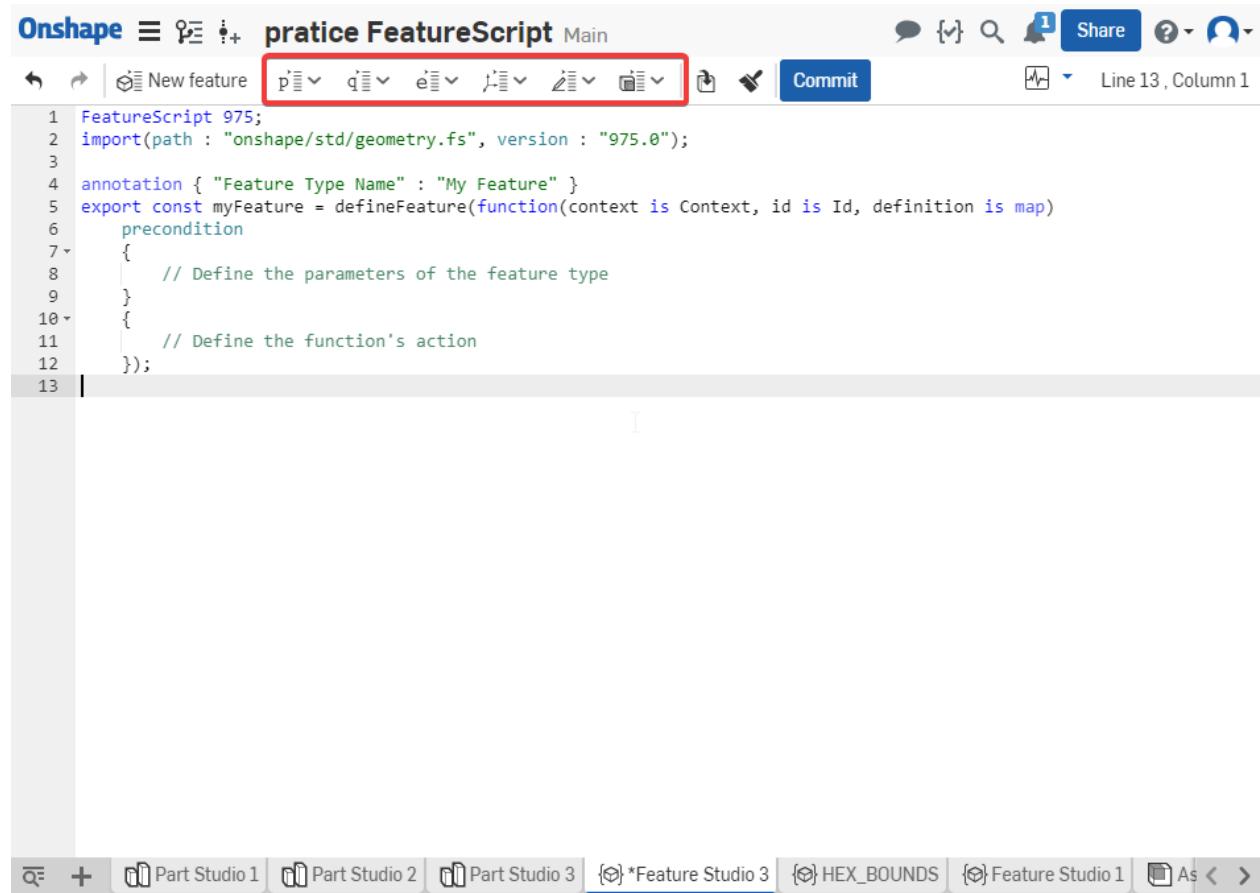


圖 6.2: 新的 Feature studio 開啟位置

6.3 建立表單介紹

快捷指令列。如圖 6.3



The screenshot shows the Onshape FeatureScript editor interface. At the top, there's a navigation bar with icons for back, forward, and search, followed by a 'Share' button and a user profile icon. Below the navigation bar is a toolbar with various icons. A red box highlights a row of icons at the top of the toolbar, which are likely keyboard shortcuts for common operations. The main area is a code editor displaying FeatureScript code:

```
1 FeatureScript 975;
2 import(path : "onshape/std/geometry.fs", version : "975.0");
3
4 annotation { "Feature Type Name" : "My Feature" }
5 export const myFeature = defineFeature(function(context is Context, id is Id, definition is map)
6   precondition
7   {
8     // Define the parameters of the feature type
9   }
10  {
11    // Define the function's action
12  });
13 |
```

At the bottom of the editor, there are tabs for different studios: Part Studio 1, Part Studio 2, Part Studio 3, Feature Studio 3 (which is currently selected), HEX_BOUNDS, Feature Studio 1, and As. There are also navigation arrows and a 'As' button.

圖 6.3: 快捷指令列

首先使用第一個 New Feature 建立新的特徵。如圖 6.4

功能主要是建立一個特徵功能的架構如下圖。如圖 6.5

可以藉由輸入程式於下圖紅框處，製作表單與繪畫功能。如圖 6.6

建立表單的各項輸入項目，可以於下圖位置點選所需項目。如圖 6.7

舉例如下圖使用第一個長度參數。如圖 6.8

使用該功能就會出現長度輸入欄。如圖 6.9

The screenshot shows the Onshape FeatureScript editor interface. At the top, there's a toolbar with various icons. Below the toolbar is a header bar with the text "Onshape" and "pratice FeatureScript Main". To the right of the header are buttons for "Share", "Help", and "User". The main area is a code editor with a syntax-highlighted script:

```
1 FeatureScript 975;
2 import(path : "onshape/std/geometry.fs", version : "975.0");
3
4
```

The first line, "New feature", is highlighted with a red rectangle. Below the code editor is a status bar showing "Line 4, Column 1". At the bottom of the screen, there's a tab bar with several tabs: "Part Studio 1", "Part Studio 2", "Part Studio 3", "Feature Studio 3" (which is currently selected), "HEX_BOUNDS", "Feature Studio 1", and "Ass".

圖 6.4: 新的特徵

Onshape practice FeatureScript Main

New feature | p q e J Z F Commit Line 3 , Column 1

```
1 FeatureScript 975;
2 import(path : "onshape/std/geometry.fs", version : "975.0");
3
4 annotation { "Feature Type Name" : "My Feature" }
5 export const myFeature = defineFeature(function(context is Context, id is Id, definition is map)
6   precondition
7   {
8     // Define the parameters of the feature type
9   }
10  {
11    // Define the function's action
12  });
13
```

Part Studio 1 Part Studio 2 Part Studio 3 *Feature Studio 3 HEX_BOUNDS Feature Studio 1 As < >

The screenshot shows the Onshape FeatureScript editor interface. The title bar says "Onshape" and "practice FeatureScript Main". Below the title bar are standard file navigation buttons and a toolbar with icons for "New feature", "Commit", and "Line 3, Column 1". The main area contains a code editor with numbered lines from 1 to 13. Lines 4 through 12 are enclosed in a red rectangular box, highlighting the feature type definition. The code itself defines a FeatureScript 975 script that imports "onshape/std/geometry.fs", defines an annotation for "My Feature", and creates a feature type "myFeature" using "defineFeature". It includes a "precondition" block and a main function block with parameters "context", "id", and "definition". The bottom of the screen shows tabs for "Part Studio 1", "Part Studio 2", "Part Studio 3", "*Feature Studio 3" (which is selected), "HEX_BOUNDS", "Feature Studio 1", and "As".

圖 6.5: 特徵格式

Onshape ≡ pratice FeatureScript Main Share

New feature | Commit Line 3 , Column 1

```
1 FeatureScript 975;
2 import(path : "onshape/std/geometry.fs", version : "975.0");
3 |
4 annotation { "Feature Type Name" : "My Feature" }
5 export const myFeature = defineFeature(function(context is Context, id is Id, definition is map)
6   precondition
7   {
8     // Define the parameters of the feature type
9   }
10  {
11    // Define the function's action
12  });
13
```

表單輸入項目程式
程式功能編寫

Part Studio 1 Part Studio 2 Part Studio 3 *Feature Studio 3 HEX_BOUNDS Feature Studio 1 As < >

圖 6.6: 特徵編輯內容

Onshape practice FeatureScript Main

New feature Commit Line 13 , Column 1

```
1 FeatureScript 975;
2 import(path : "ons
3
4 annotation { "Feat
5 export const myFea
6 precondition
7 {
8     // Define
9 }
10 {
11     // Define
12 });
13
```

The parameter dropdown menu is open, showing the following options:

- Length parameter
- Angle parameter
- Count parameter
- Query parameter
- Enum parameter
- Boolean parameter
- String parameter
- Array parameter
- Selection-driven array parameter
- Part Studio reference parameter

Bottom navigation bar: Part Studio 1, Part Studio 2, Part Studio 3, *Feature Studio 3 (selected), HEX_BOUNDS, Feature Studio 1, As < >

圖 6.7: 插入表單

A screenshot of a FeatureScript editor interface. The code being edited is:

```
1 FeatureScript 975;
2 import(path : "onsl
3
4
5 annotation { "Feat
6 export const myFeat
7   precondition
8   {
9     annotation
10    isLength(d
11
12  }
13  {
14
15 });
16
```

The word "Length parameter" is highlighted with a red box. A context menu is open over this word, listing various parameter types. The menu items are:

- Length parameter
- Angle parameter
- Count parameter
- Query parameter
- Enum parameter
- Boolean parameter
- String parameter
- Array parameter
- Selection-driven array parameter
- Part Studio reference parameter
- Image reference parameter
- CSV reference parameter

圖 6.8: Length

A screenshot of a feature configuration dialog. The title bar says "My Feature 1". There are two buttons: a green checkmark and a red X. Below the title is a table with one row:

My Length	25 mm
-----------	-------

Below the table is a horizontal slider with a circular track and a small circle indicating the current value.

圖 6.9: 長度表單

6.4 自製繪圖功能練習

成果圖: 如圖 6.10

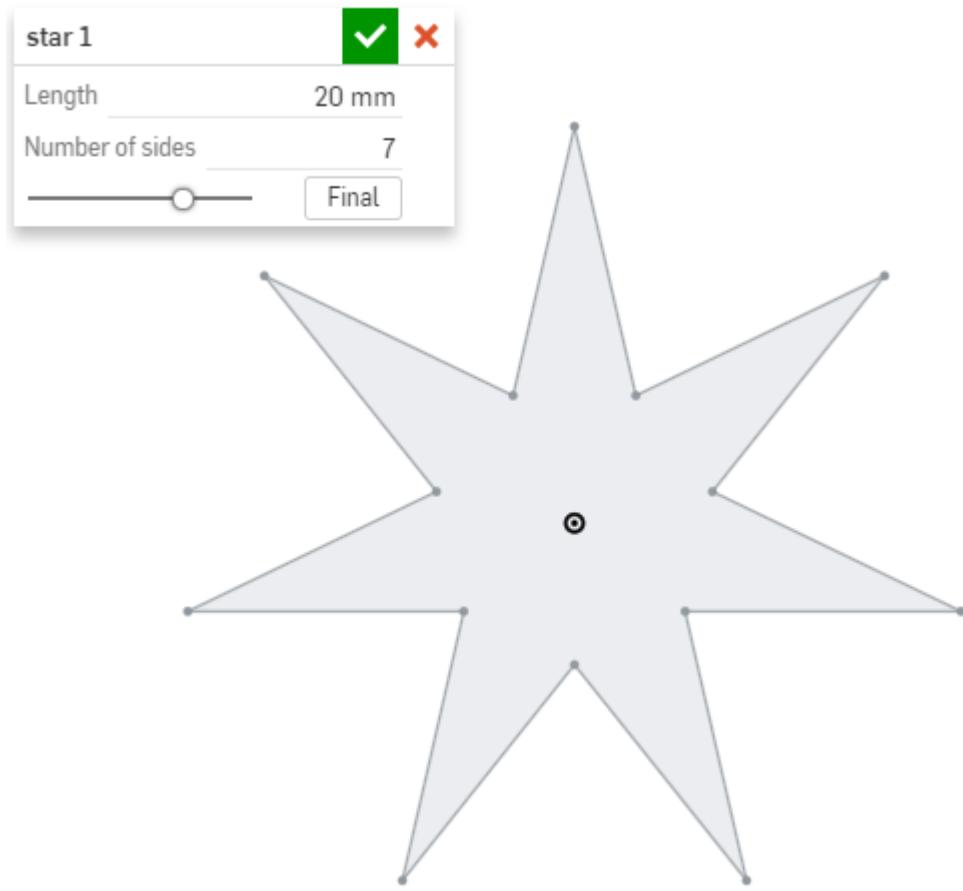


圖 6.10: 星星練習成果

原 始 碼:<https://mdecadp2018.github.io/site-40623128/content/%E8%87%AA%E8%A3%BD%E7%B9%AA%E5%9C%96%E5%8A%9F%E8%83%BD%E7%B7%B4%E7%BF%92-%E6%98%9F%E6%98%9F.html>

六角填充註解:<https://mdecadp2018.github.io/site-40623128/content/%E5%85%AD%E8%A7%92%E5%A1%AB%E5%85%85.html>

第七章 參考文獻

Remote API Functions (Python):

<http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm>

Onshape Featurescript:

<https://www.onshape.com/features/custom-features>

Onshape 內建功能原始碼:

<https://cad.onshape.com/documents/12312312345abcabcabcdeff>

Featurescript 教學:

<https://cad.onshape.com/FsDoc/index.html>

<https://cadlab.mde.tw/post/tag/featurescript.html>

蜂巢填充:

<https://cad.onshape.com/documents/84792bd1b9878b2081e6fa18/w/78e203b9d630dc6a62b7be58/e/d54e2d4096642b5acf95c1f7>