บทที่ 5

การออกแบบโปรแกรม

ในบทที่ 2 ถึงบทที่ 4 ได้กล่าวถึงทฤษฎีและความรู้พื้นฐานในการศึกษาและทำวิจัย สำหรับบทนี้จะกล่าวถึงแนวคิดในการออกแบบการทำงานของโปรแกรม โครงสร้างข้อมูล และส่วนประกอบที่สำคัญของโปรแกรม

การออกแบบการทำงานของโปรแกรม

ผู้ทำการวิจัย ได้ทำการออกแบบโปรแกรมการวิเคราะห์ภาพถ่ายรังสีเอกซ์ ของกระโหลกศีรษะ โดยยึดหลักของการออกแบบโปรแกรมเชิงวัตถุ โดยทำการกำหนดวัตถุ (Object) ขึ้นจำนวน 9 วัตถุ ดังมีชื่อและหน้าที่ ดังต่อไปนี้

1. CCepApp

เป็นวัตถุที่เป็นส่วนตัวโปรแกรมหลัก ของโปรแกรมบนวินโดว์ โดยมีหน้าที่ในการเริ่มต้นโปรแกรม คอยรับและประมวลผลข้อความ (message) ที่ได้รับจากวินโดว์อื่น ๆ และส่งข้อความ (message) ให้กับวินโดว์ลูก หรือวินโดว์อื่น ๆ

2. CMainFrame

เป็นวัตถุที่เป็นส่วนโครงของโปรแกรมโดยจะประกอบด้วยส่วนเมนู, ส่วนแถบแสดงสถานะ, ส่วนแถบเครื่องมือ และส่วนต้นแบบเอกสาร

3. CCepDoc

เป็นวัตถุที่เป็นส่วนจัดการกับข้อมูล โดยใช้ในการเรียกข้อมูล เปลี่ยนแปลงข้อมูล จัดเก็บข้อมูล และเป็นส่วนให้ส่วนมุมมองใช้ในการติดต่อกับข้อมูลโดยทางอ้อม

4. CCepView

เป็นวัตถุที่เป็นมุมมองเพื่อใช้ในการแสดงภาพของกระโหลกศีรษะในลักษณะโครงร่าง โดยผู้ใช้สามารถสั่งให้ CCepView ทำการแสดงภาพเฉพาะกระโหลกศีรษะ หรือแสดงภาพกระโหลกศีรษะพร้อมการวิเคราะห์ตามวิธีที่กล่าวมาแล้วในบทที่ 4

5. CResultVw

เป็นวัตถุที่เป็นมุมมองเพื่อใช้ในการแสดงผลการวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะ ในลักษณะตัวเลขแสดงมุมและระยะทาง ตามเกณฑ์การวิเคราะห์ในบทที่ 4

6. CNoteVw

เป็นวัตถุที่เป็นมุมมองเพื่อใช้ในการบันทึกข้อมูล ในลักษณะตัวอักษร โดยจะติดต่อกับ CCepDoc เพื่อให้บันทึกตัวอักษรที่ผู้ใช้พิมพ์

7. CDigView

เป็นวัตถุที่เป็นมุมมองเพื่อใช้ในการแสดงภาพระหว่างการบันทึกจุดโดยเครื่องอ่านพิกัด โดย CdigView จะติดต่อกับ CCepDoc เพื่อให้บันทึกค่าของจุดลงใน CCepDoc

8. CGrowthVw

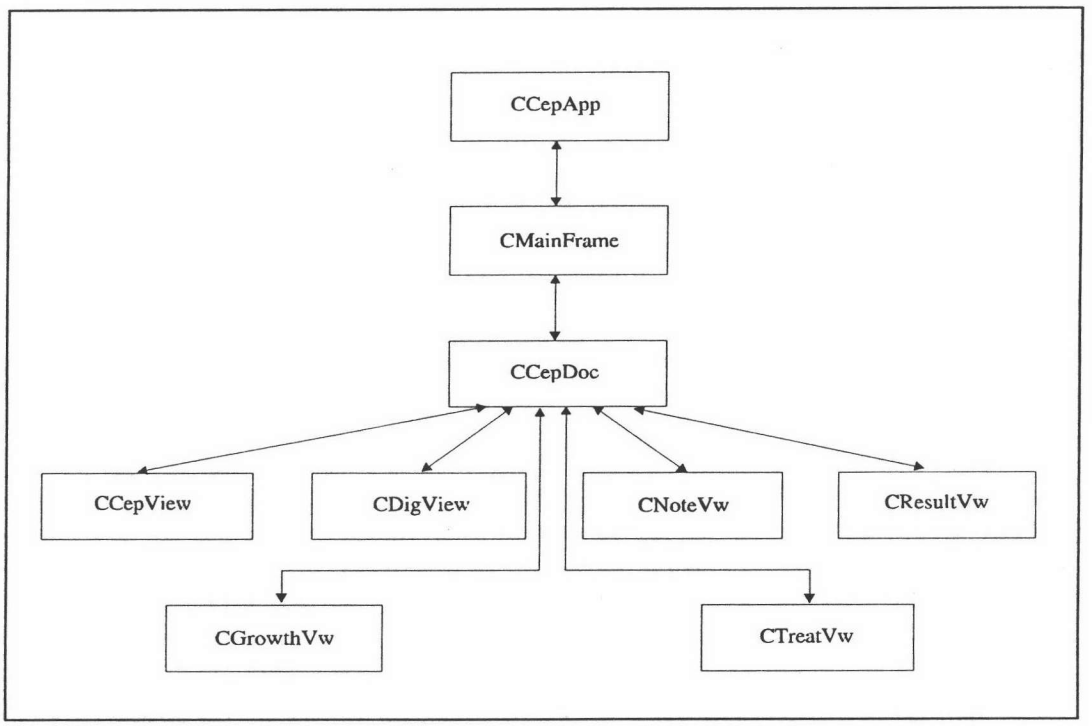
เป็นวัตถุที่เป็นมุมมองเพื่อใช้ในการแสดงภาพการเปลี่ยนแปลงของกระโหลกศีรษะ เนื่องจากการเปลี่ยนแปลงของอายุของผู้ป่วย โดยผู้ใช้สามารถกำหนดค่าอายุที่เพิ่มขึ้นของผู้ป่วยผ่าน CGrowthVw

9. CTreatVw

เป็นวัตถุที่เป็นมุมมองเพื่อใช้ในการแสดงภาพการเปลี่ยนแปลงของกระโหลกศีรษะเนื่องจากการผ่าตัด โดยผู้ใช้สามารถใช้เมาส์ในการเลือกชิ้นของกระดูกที่จะทำการเปลี่ยนแปลง

โดยสามารถเปรียบเทียบกับโครงสร้างมาตรฐานตามที่กล่าวแล้วในบทที่ 3 รูปที่ 3.5 ดังนี้ วัตถุ CCepApp คือส่วนตัวโปรแกรม วัตถุ CMainFrame คือส่วนโครงโปรแกรม วัตถุ CCepDoc คือส่วนเอกสาร และประกอบด้วยมุมมอง 6 มุมมองซึ่งมีความสัมพันธ์กับส่วนเอกสารคือ วัตถุ CCepView วัตถุ CResultVw วัตถุ CNoteVw วัตถุ CDigView วัตถุ

CGrowthVw และวัตถุ CTreatVw และสามารถแสดงความสัมพันธ์ของวัตถุทั้ง 9 ชิ้นดังแสดงในรูปที่ 5.1



รูปที่ 5.1 แสดงความสัมพันธ์ระหว่างวัตถุของโปรแกรมวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะ

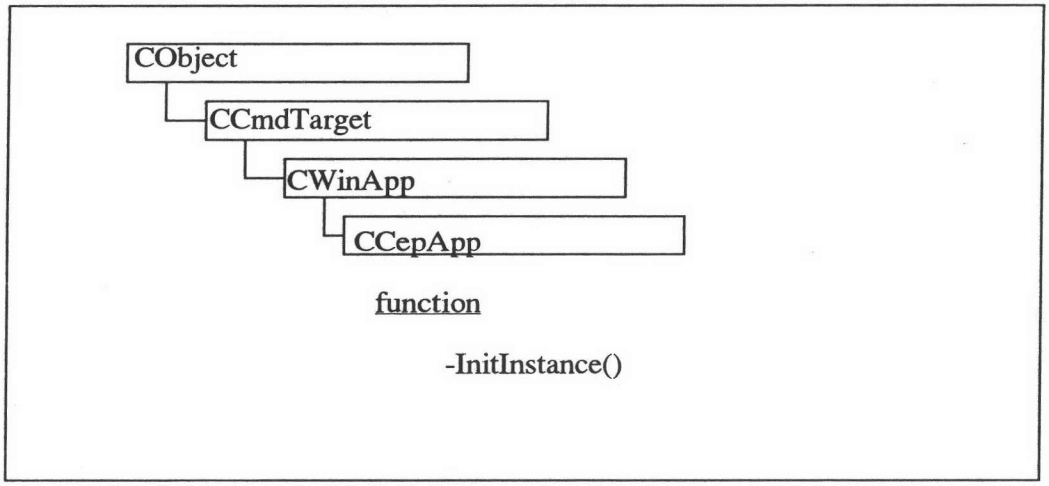
<u>โครงสร้างข้อมูล</u>

ตามที่กล่าวแล้วเบื้องต้น ผู้ทำการวิจัยได้ออกแบบโปรแกรมวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะ โดยยึดหลักของการออกแบบโปรแกรมเชิงวัตถุ ผู้ทำการวิจัยได้เลือกใช้ภาษา C++ ในการพัฒนาโปรแกรม เนื่องจากภาษา C++ มีลักษณะเป็นภาษาผสม ที่สามารถเขียนโปรแกรมในลักษณะโปรแกรมเชิงวัตถุ หรือเขียนโปรแกรมในลักษณะของภาษา C แบบดั้งเดิมได้

ในหัวข้อนี้ ผู้ทำการวิจัยจะกล่าวถึงโครงสร้างข้อมูลของวัตถุ ทั้ง 9 วัตถุ โดยสังเขป และใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล

1. CCepApp

CCepApp เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส CWinApp และทำการทับฟังก์ชัน InitInstance( ) ของ CWinApp ดังแสดงในรูปที่ 5.2 โดยฟังก์ชัน InitInstance จะเป็นตัวกำหนดค่าเริ่มต้นของโปรแกรม



รูปที่ 5.2 แสดงคลาส CCepApp

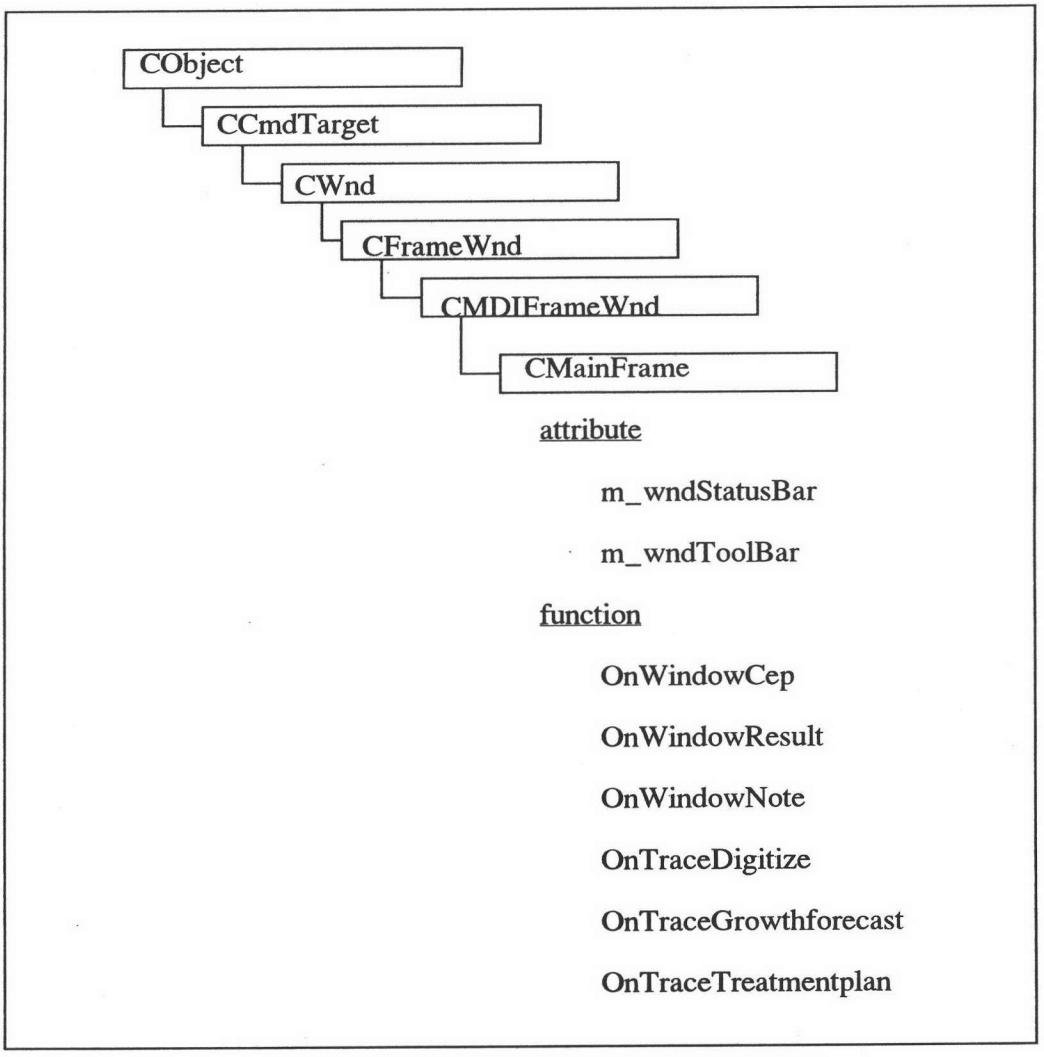
และสามารถแสดงโครงสร้างข้อมูลโดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.3

```
class CCepApp : public CWinApp
{
Public :
        CCepApp ( ) ;
// Overrides
        Virtual BOOL InitInstance ( ) ;
}
```

รูปที่ 5.3 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CCepApp

2. CMainFrame

CMainFrame เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส CMDIFrameWnd ดังแสดงในรูปที่ 5.4



รูปที่ 5.4 แสดงคลาส CMainFrame

คลาส CMainFrame ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลประกอบด้วยข้อมูล 2 ชนิดคือ m_wndStatusbar เป็นส่วนแถบแสดงสถานะ และ m_wndToolBar เป็นส่วนแถบเครื่องมือ

และส่วนฟังก์ชันประกอบด้วยฟังก์ชันดังนี้

OnwindowCep เป็นฟังก์ชันในการสร้างวัตถุ CCepview

OnWindowResult เป็นฟังก์ชันในการสร้างวัตถุ CResultVw

OnWindowNote เป็นฟังก์ชันในการสร้างวัตถุ CNoteVw

OnTraceDigitize เป็นฟังก์ชันในการสร้างวัตถุ Cdigview

OnTraceGrowthforecast เป็นฟังก์ชันในการสร้างวัตถุ Cgrowthvw

OnTraceTreatmentplan เป็นฟังก์ชันในการสร้างวัตถุ CTreatvw

สามารถแสดงโครงสร้างข้อมูลของ CMainFrameโดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.5

```
class CMainFrame : public CMDIFrameWnd
{
        DECLARE_DYNAMIC(CMainFrame)
public:
        CMainFrame();
        virtual ~CMainFrame();
protected:      // control bar embedded members
        CStatusBar      m_wndStatusBar;
        CToolBar        m_wndToolBar;
// Generated message map functions
protected:
```

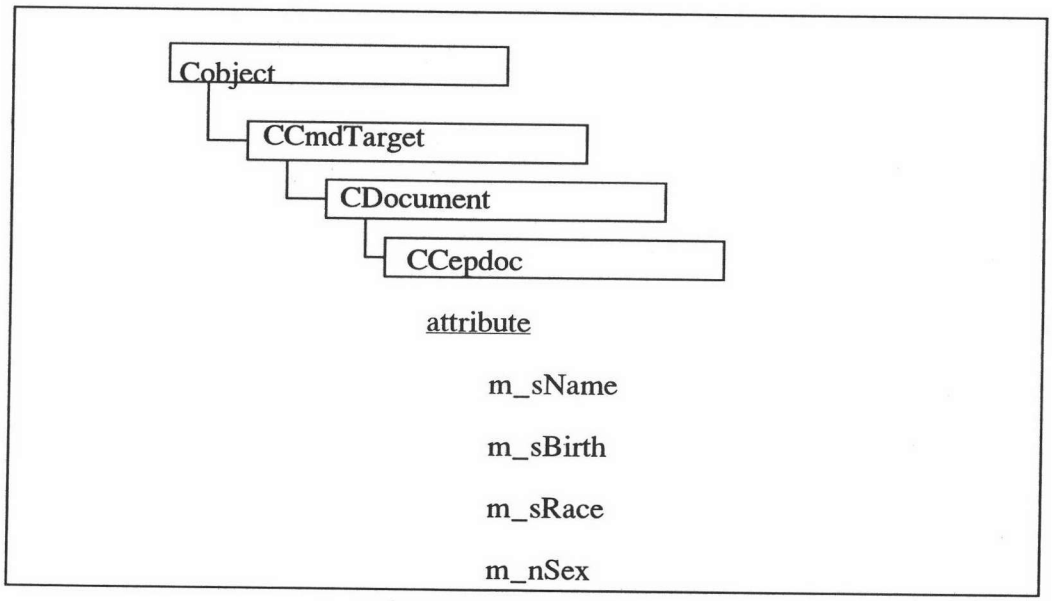
รูปที่ 5.5 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CMainFrame

```
        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        afx_msg void OnWindowCep();
        afx_msg void OnWindowResult();
        afx_msg void OnWindowNote();
        afx_msg void OnTraceDigitize();
        afx_msg void OnTraceGrowthforcast();
        afx_msg void OnTraceTreatmentplan();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

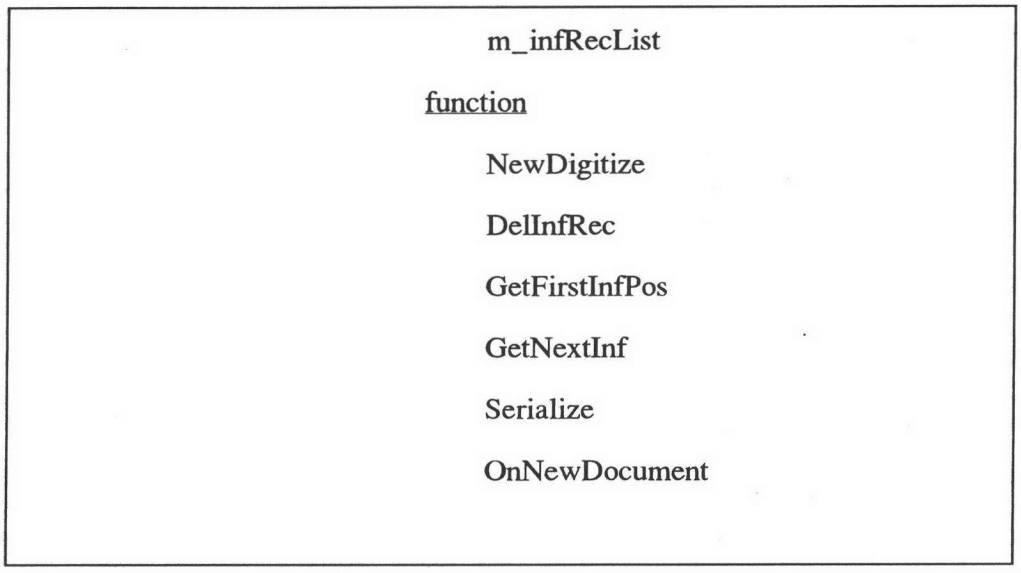
รูปที่ 5.5 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CMainFrame (ต่อ)

3. CCepDoc

CCepDoc เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส CDocument ดังแสดงในรูปที่ 5.6



รูปที่ 5.6 แสดงคลาส CCepDoc



รูปที่ 5.6 แสดงคลาส CCepDoc (ต่อ)

คลาส CCepDoc ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลประกอบด้วยข้อมูล 5 ชนิดคือ m_sName เป็นส่วนข้อมูลชื่อผู้ป่วย m_sBirth เป็นส่วนข้อมูลวันเกิด m_sRace เป็นส่วนข้อมูลเชื้อชาติ m_nSex เป็นส่วนข้อมูลเพศ และ m_infRecList เป็นแถวลำดับที่ใช้เก็บชุดข้อมูลที่บันทึกแต่ละครั้ง

และส่วนฟังก์ชันประกอบด้วยฟังก์ชันดังนี้

NewDigitize เป็นฟังก์ชันในการสร้างชุดข้อมูลใหม่

DelInfRec เป็นฟังก์ชันในการลบชุดข้อมูลปัจจุบัน

GetFirstInfPos เป็นฟังก์ชันในการหาชุดข้อมูลชุดแรก

GetNextDig เป็นฟังก์ชันในการหาชุดข้อมูลชุดต่อไป

Serialize เป็นฟังก์ชันในการเก็บรักษาข้อมูล

OnNewDocument เป็นฟังก์ชันในการสร้างเอกสารชุดใหม่

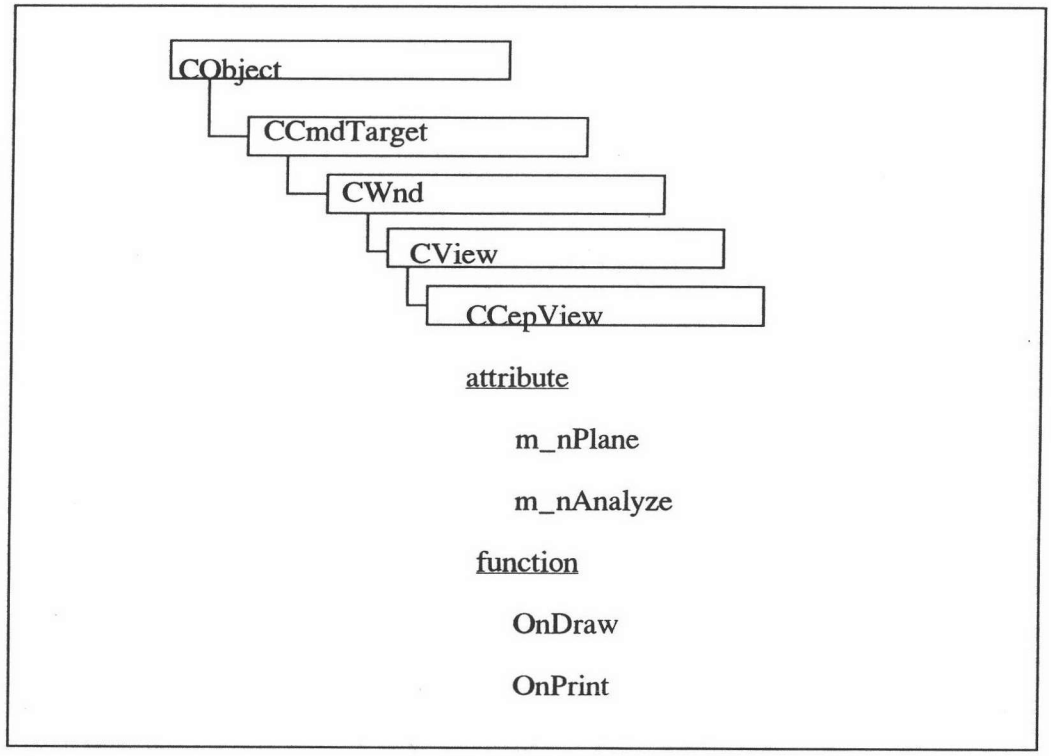
และสามารถแสดงโครงสร้างข้อมูลของ CCepDoc โดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.7

```
class CCepdoc : public CDocument
{
protected: // create from serialization only
        CCepDoc();
        DECLARE_DYNCREATE(CCepDoc)
// Attributes
protected:
        CString         m_sName;      // Name of patient
        CString         m_sBirth;     // Birthday of patient
        CString         m_sRace;      // Race of patient
        int             m_nSex;
        CObList         m_infRecList; // List of information in
                                         each record
public:
        CDigPoints* NewDigitize();
        void DelInfRec();
        POSITION GetFirstInfPos();
        CInfRec* GetNextInf(POSITION& pos);
        virtual ~CCepDoc();
        virtual void Serialize(CArchive& ar);
        virtual  BOOL  OnNewDocument();
};
```

รูปที่ 5.7 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CCepDoc

4. CCepView

CCepView เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส Cview ดังแสดงในรูปที่ 5.8



รูปที่ 5.8 แสดงคลาส CCepView

คลาส CCepView ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลประกอบด้วยข้อมูล 2 ชนิดคือ m_nPlane เป็นส่วนข้อมูลการเลือกระนาบที่ใช้ในการแสดงผล และ m_nAnalyze เป็นส่วนข้อมูลการเลือกชนิดการวิเคราะห์

ส่วนฟังก์ชันประกอบด้วยฟังก์ชันสองฟังก์ชันคือ ฟังก์ชัน OnDraw เป็นฟังก์ชันในการแสดงผลออกทางคอนเทกซ์ (Context) อาจเป็นทางจอภาพหรือทางเครื่องพิมพ์ ฟังก์ชัน OnPrint เป็นฟังก์ชันเพื่อใช้ในการพิมพ์

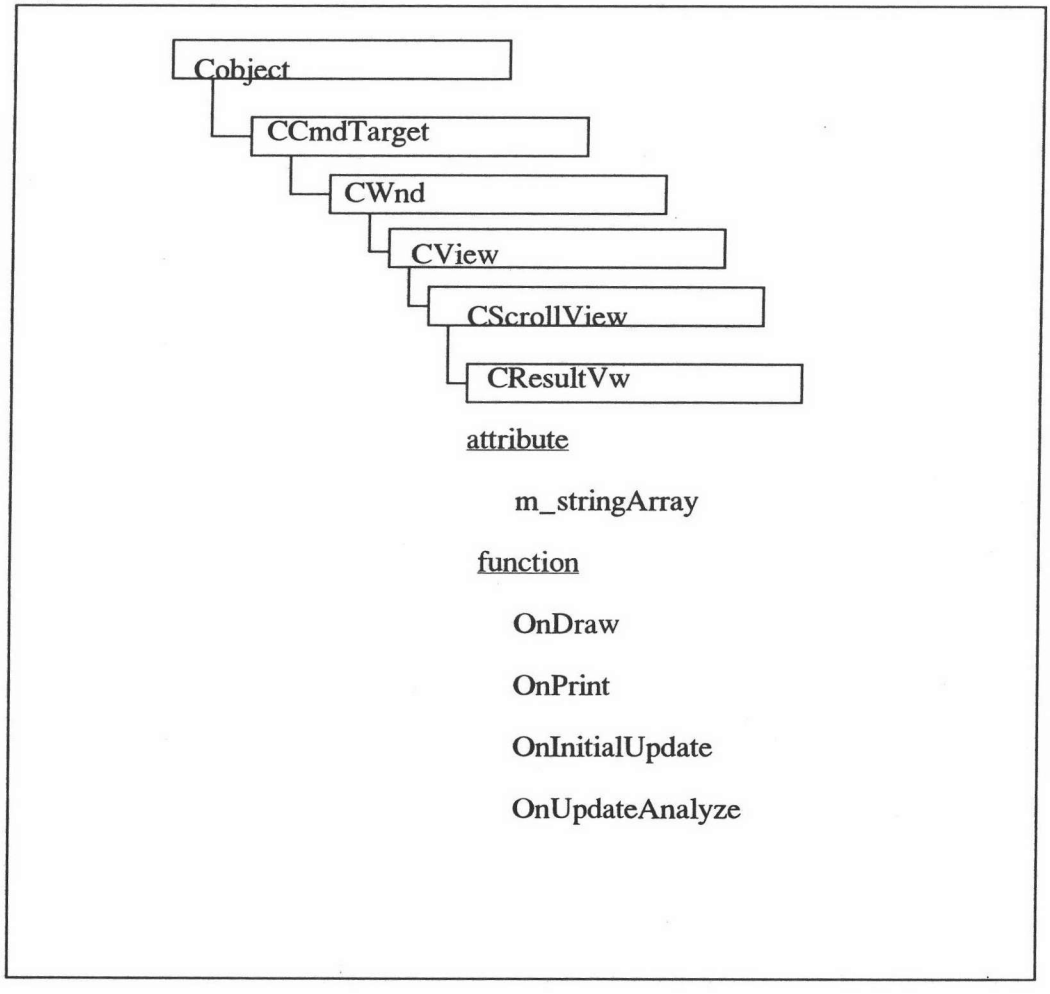
สามารถแสดงโครงสร้างข้อมูลของ CCepView โดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.9

```
class CCepView : public CView
{
protected: // create from serialization only
        CCepView();
        DECLARE_DYNCREATE(CCepView)
// Attributes
public:
        CCepDoc* GetDocument();
protected:
        UINT            m_nPlane;       // Type of Plane.
        UINT            m_nAnalyze;     // Type of Analyze.
// Implementation
public:
        virtual ~CCepView();
        virtual void OnDraw(CDC* pDC);  // overridden to draw this view
        virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
}
```

รูปที่ 5.9 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CCepView

5. CResultVw

CResultVw เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส CScrollView ดังแสดงในรูปที่ 5.10



รูปที่ 5.10 แสดงคลาส CCepView

คลาส CResultVw ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลมีข้อมูล m_stringArray ซึ่งเป็นแถวลำดับที่ใช้เก็บข้อมูลผลการวิเคราะห์

และประกอบด้วยส่วนฟังก์ชันดังนี้

OnDraw เป็นฟังก์ชันในการแสดงผลออกทางคอนเทกซ์ (Context) อาจเป็นทางจอภาพหรือทางเครื่องพิมพ์

OnPrint เป็นฟังก์ชันเพื่อใช้ในการพิมพ์

OnInitialUpdate เป็นฟังก์ชันที่ถูกเรียกหลังจากติดต่อกับส่วนเอกสาร แต่ก่อนการแสดงผลทางจอภาพ

OnUpdateAnalyze เป็นฟังก์ชันใช้ในการคำนวณผลการวิเคราะห์

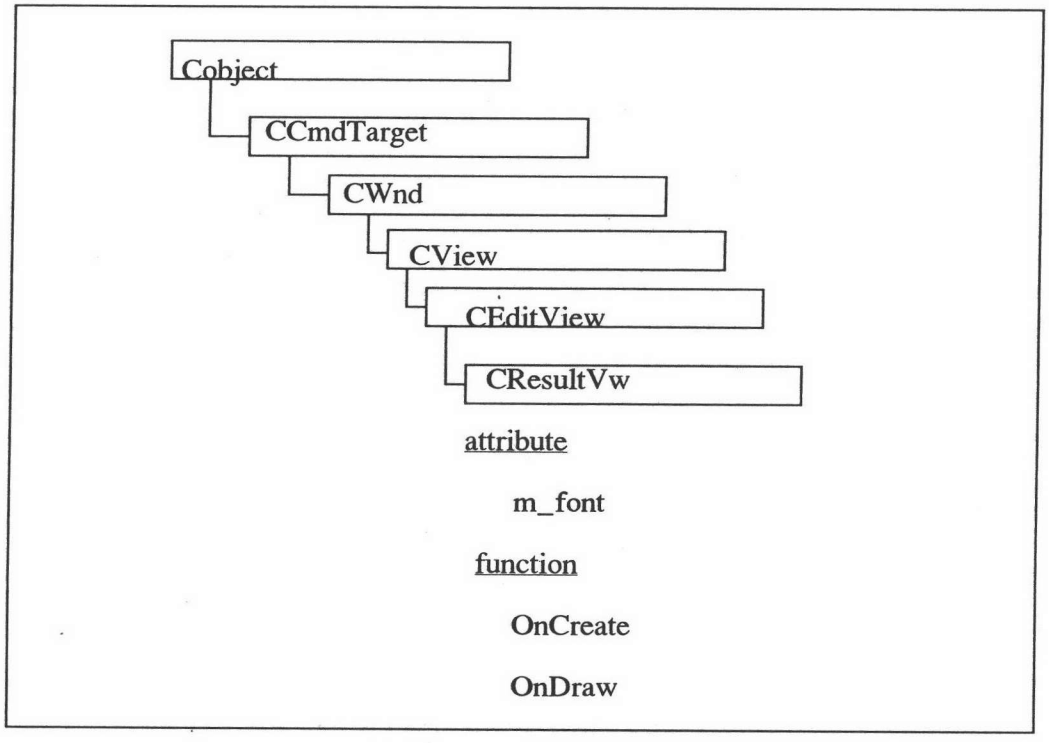
และสามารถแสดงโครงสร้างข้อมูลของ CResultVw โดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.11

```
class CResultVw : public CScrollView
{
        DECLARE_DYNCREATE(CResultVw)
protected:
        CResultVw();
// Attributes
public:
        CStringArray m_stringArray;  // Array of Result String
// Implementation
public:
        virtual void OnPrint(CDC* pDC, CPrintInfo* pInfo);
protected:
        virtual ~CResultVw();
        virtual  void OnDraw(CDC* pDC);   // overridden to draw this view
        virtual  void OnInitialUpdate();        // first time after construct
        void UpdateAnalyze();                   // re-analyze all data
}
```

รูปที่ 5.11 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CResultVw

6. CNoteVw

CNoteVw เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส CEditView ดังแสดงในรูปที่ 5.12



รูปที่ 5.12 แสดงคลาส CNoteVw

คลาส CNoteVw ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลมีข้อมูล m_font เป็นส่วนที่ใช้ในการเก็บแบบอักษร

ส่วนฟังก์ชันมีฟังก์ชัน OnCreate ซึ่งใช้ในการสร้างแบบอักษร และฟังก์ชัน OnDraw ซึ่งใช้ในการแสดงผลออกทางคอนเทกซ์

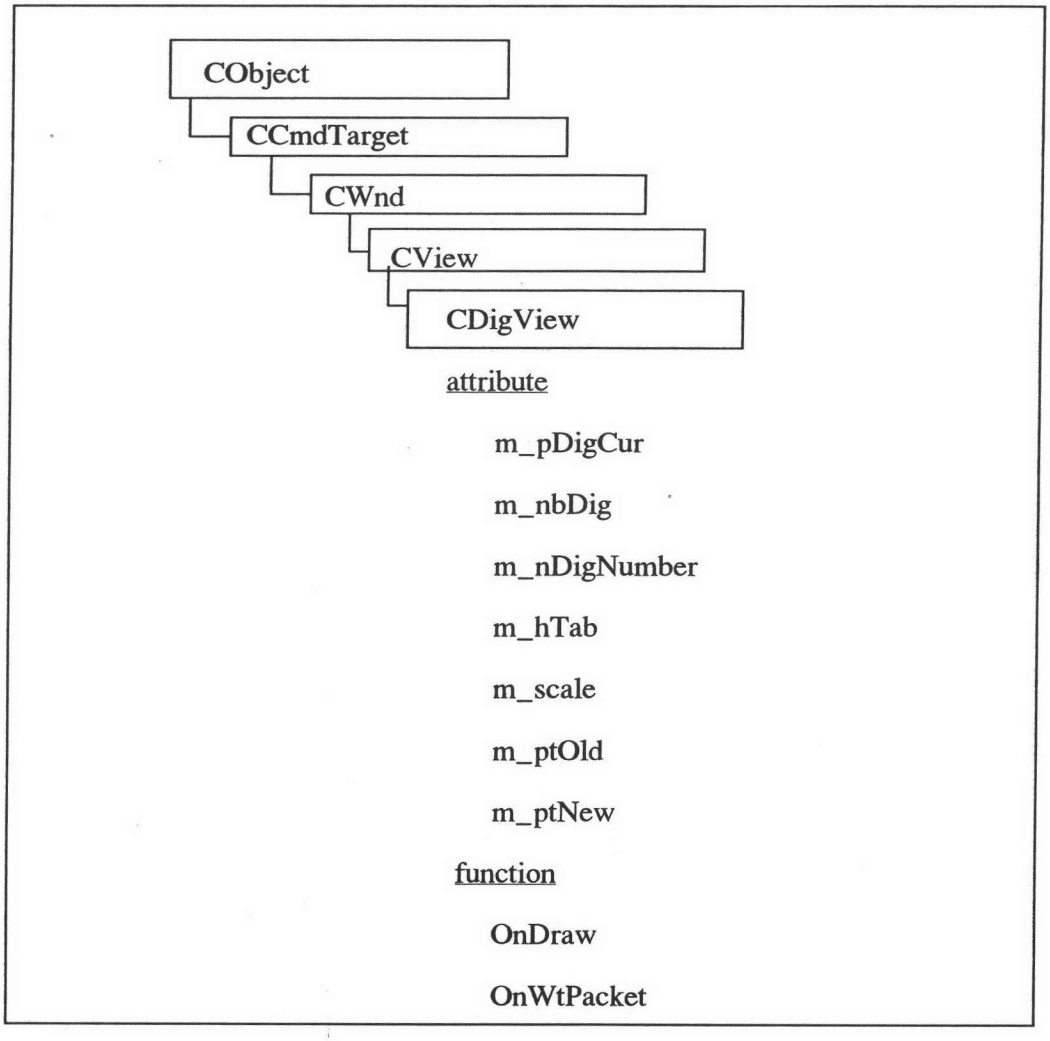
และสามารถโครงสร้างข้อมูลของ CNoteVw โดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.13

```
class CNoteVw : public CEditView
{
        DECLARE_DYNCREATE(CNoteVw)
protected:
        CNoteVw();          // protected constructor used by dynamic creation
// Attributes
public:
        CFont m_font;
// Implementation
protected:
        virtual ~CNoteVw();
        virtual  void OnDraw(CDC* pDC);   // overridden to draw this view
        // Generated message map functions
protected:
        //{{AFX_MSG(CNoteVw)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
}
```

รูปที่ 5.13 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CNoteVw

7. CDigView

CDigView เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส Cview ดังแสดงในรูปที่ 5.14



รูปที่ 5.14 แสดงคลาส CDigView

คลาส CDigView ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลประกอบด้วยข้อมูล 7 ชนิดคือ m_pDigCur เป็นส่วนข้อมูลตัวชี้ไปยังวัตถุ CDigPoints m_bDig เป็นส่วนข้อมูลแสดงการเริ่มต้นรับข้อมูลพิกัดของจุด m_nDigNumber เป็นส่วนข้อมูลแสดงเลขที่ของจุดปัจจุบัน m_hTab เป็นส่วนข้อมูลคอนเทกซ์ของเครื่อง

อ่านพิกัด m_scale เป็นส่วนข้อมูลแสดงอัตราความละเอียดของเครื่องอ่านพิกัด m_ptOld เป็นข้อมูลตำแหน่งเดิมของเครื่องอ่านพิกัด และ m_ptNew คือข้อมูลตำแหน่งที่เปลี่ยนแปลง

ส่วนฟังก์ชันประกอบด้วยฟังก์ชันสองฟังก์ชันคือ ฟังก์ชัน OnDraw เป็นฟังก์ชันในการแสดงผลออกทางคอนเทกซ์ (Context) อาจเป็นทางจอภาพหรือทางเครื่องพิมพ์ และฟังก์ชัน OnWTPacket เป็นฟังก์ชันสำหรับรับข้อความที่ส่งจากโปรแกรมขับของเครื่องอ่านพิกัด

สามารถแสดงโครงสร้างข้อมูลของ CDigView โดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.15

```
class CDigView : public CView
{
        DECLARE_DYNCREATE(CDigView)
protected:
        CDigView();   // protected constructor used by dynamic creation
// Attributes
protected:
        CDigPoints*         m_pDigCur;    // Cureent Digitize picture
        BOOL                m_bDig;       // Start Digitize flag
        UINT                m_nDigNumber; // Cureent Number of
                                                Digitize point
        HCTX                m_hTab;       // Tablet context
        FIX32               m_scale[2];   // Tablet Scale
        POINT               m_ptOld;      // Previous Digitize point
        POINT               m_ptNew;      // New Digitize point
```

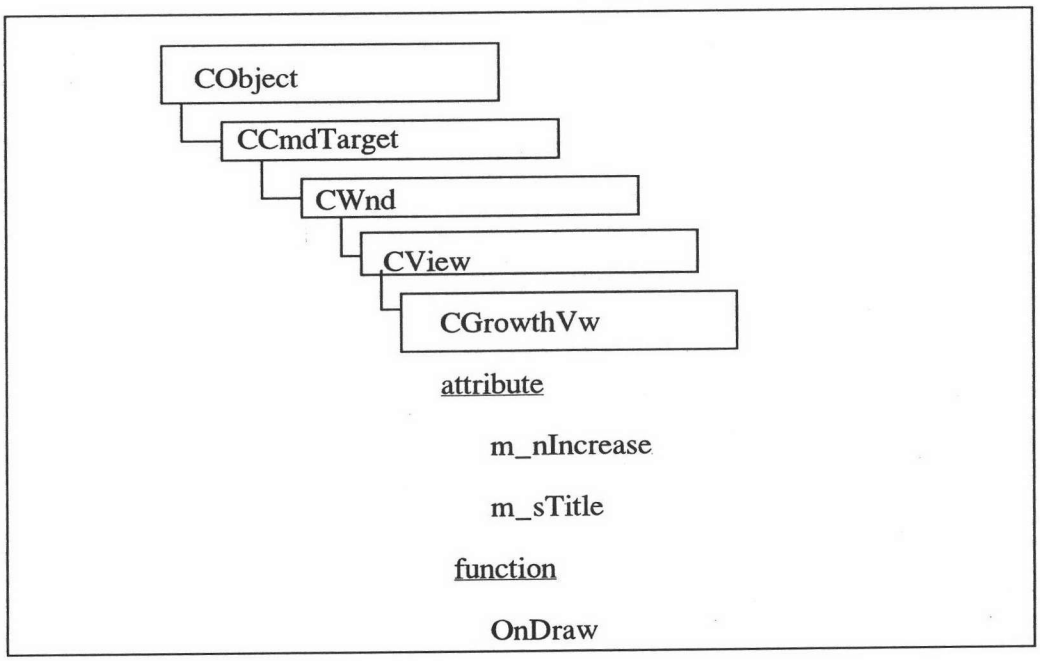
รูปที่ 5.15 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CDigView

```
        virtual  void OnDraw(CDC* pDC);   // overridden to draw this view
        // Generated message map functions
protected:
        //{{AFX_MSG(CDigView)
        afx_msg LRESULT OnWTPacket(WPARAM wParam,
                                   LPARAM lParam);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
}
```

รูปที่ 5.15 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CDigView (ต่อ)

8. CGrowthVw

CGrowthVw เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส Cview ดังแสดงในรูปที่ 5.16



รูปที่ 5.16 แสดงคลาส CGrowthVw

คลาส CGrowthVw ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลประกอบด้วยข้อมูลสองชนิดคือ m_nIncrease เป็นส่วนข้อมูลจำนวนปีที่เพิ่มขึ้นเพื่อใช้ในการทำนายการเปลี่ยนแปลงของกระโหลกศีรษะ และ m_sTitle เป็นส่วนข้อมูลที่ใช้เก็บชื่อเดิมของวินโดว์ ส่วนฟังก์ชันประกอบด้วยฟังก์ชัน OnDraw ซึ่งเป็นฟังก์ชันในการแสดงผลออกทางคอนเท็กซ์ (Context) อาจเป็นทางจอภาพหรือทางเครื่องพิมพ์

สามารถแสดงโครงสร้างข้อมูลของ CGrowthVw โดยใช้ภาษา C++ ได้ดังแสดงในรูปที่ 5.17

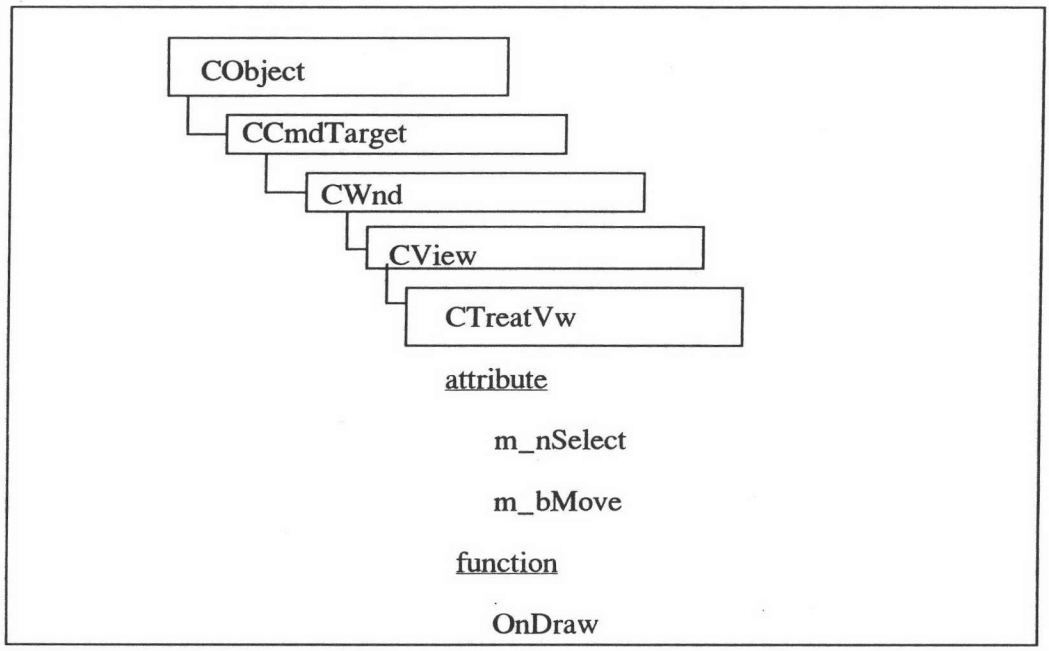
```
class CGrowthVw : public CView
{
        DECLARE_DYNCREATE(CGrowthVw)
protected:
        CGrowthVw();      // protected constructor used by dynamic creation
// Attributes
public:
        UINT  m_nIncrease;
        CString m_sTitle;

// Implementation
protected:
        virtual ~CGrowthVw();
        virtual  void OnDraw(CDC* pDC);    // overridden to draw this view
        DECLARE_MESSAGE_MAP()
}
```

รูปที่ 5.17 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CGrowthVw

9. CTreatVw

CTreatVw เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส Cview ดังแสดงใน รูปที่ 5.18



รูปที่ 5.18 แสดงคลาส CTreatVw

คลาส CTreatVw ประกอบด้วยสองส่วนคือ ส่วนข้อมูลและส่วนฟังก์ชัน โดยส่วนข้อมูลประกอบด้วยข้อมูลสองชนิดคือ m_n Select เป็นส่วนข้อมูลจำนวนที่เก็บหมายเลขชิ้นกระดูกที่เลือกเพื่อใช้ในการจำลองการเปลี่ยนแปลงของกระโหลกศีรษะ และ m_bMove เป็นส่วนข้อมูลที่ใช้เก็บสภาวะการเคลื่อนที่ของชิ้นกระดูก ส่วนฟังก์ชันประกอบด้วยฟังก์ชัน OnDraw ซึ่งเป็นฟังก์ชันในการแสดงผลออกทางคอนเทกซ์ (Context) อาจเป็นทางจอภาพหรือทางเครื่องพิมพ์

สามารถแสดงโครงสร้างข้อมูลของ CGrowthVw โดยใช้ภาษา C++ ได้ดังแสดงใน รูปที่ 5.19

```
class CTreatVw : public CView
{
        DECLARE_DYNCREATE(CTreatVw)
protected:
        CTreatVw();        // protected constructor used by dynamic creation
// Attributes
public:
// Implementation
protected:
        virtual ~CTreatVw();
        virtual  void OnDraw(CDC* pDC);   // overridden to draw this view
```

รูปที่ 5.19 แสดงการใช้ภาษา C++ ในการแสดงโครงสร้างของข้อมูล CTreatVw

ส่วนประกอบที่สำคัญของโปรแกรม

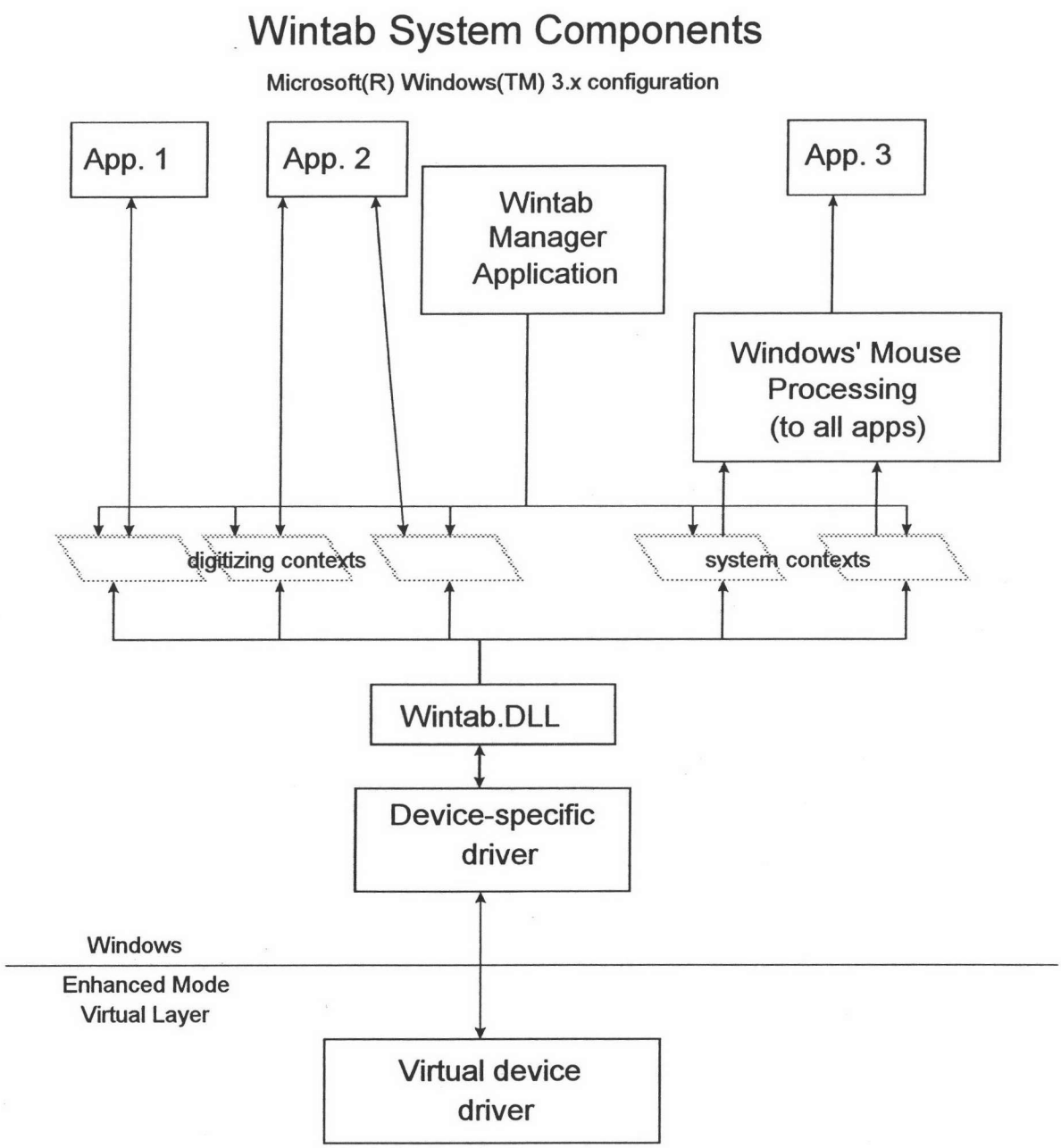
ตามหัวข้อการออกแบบการทำงานของโปรแกรม และหัวข้อโครงสร้างข้อมูลที่กล่าวมาแล้วเบื้องต้น ผู้ทำการวิจัยได้กล่าวถึงการออกแบบโปรแกรมในลักษณะของวัตถุ โดยอธิบายถึงคุณสมบัติและความสัมพันธ์ระหว่างวัตถุต่าง ๆ ในโปรแกรม ในหัวข้อนี้ ผู้ทำการวิจัยจะกล่าวถึงส่วนประกอบของโปรแกรมในลักษณะของการทำงานตามหน้าที่ โดยสามารถจำแนกตามหน้าที่ได้ 9 ส่วน ดังมีรายละเอียดดังต่อไปนี้

1. ส่วนการติดต่อกับเครื่องอ่านพิกัด

การติดต่อกับเครื่องอ่านพิกัดแต่เดิมนั้นมีข้อจำกัดเป็นอย่างมาก เนื่องจากเครื่องอ่านพิกัดที่ผลิตจากบริษัทต่าง ๆ มีข้อแตกต่างกัน จึงทำให้ผู้เขียนโปรแกรมต้องผลิตโปรแกรมมาใช้เฉพาะสำหรับเครื่องอ่านพิกัดในแต่ละบริษัท จนกระทั่งในปี ค.ศ.1991 กลุ่ม บริษัทผู้ผลิตเครื่องอ่านพิกัดได้ร่วมกันจัดประชุม และมอบให้บริษัท LCS/Telegraphics เป็นผู้กำหนดมาตรฐานการติดต่อกับเครื่องอ่านพิกัด

บริษัท LCS/Telegraphics ได้เสนอมาตรฐานที่ชื่อว่า Wintab สำหรับเครื่องอ่านพิกัด และทำการปรับปรุงจากข้อคิดเห็นของบริษัทอื่น ๆ จนได้พัฒนาเป็น Wintab API ซึ่งสามารถใช้ในการพัฒนาโปรแกรมบน Microsoft Window 3.1 และ Microsoft Window NT ซึ่งปัจจุบันบริษัทผู้ผลิตเครื่องอ่านพิกัดจำนวนมากกว่า 30 บริษัทได้ใช้ Wintab เป็นมาตรฐานในการผลิต

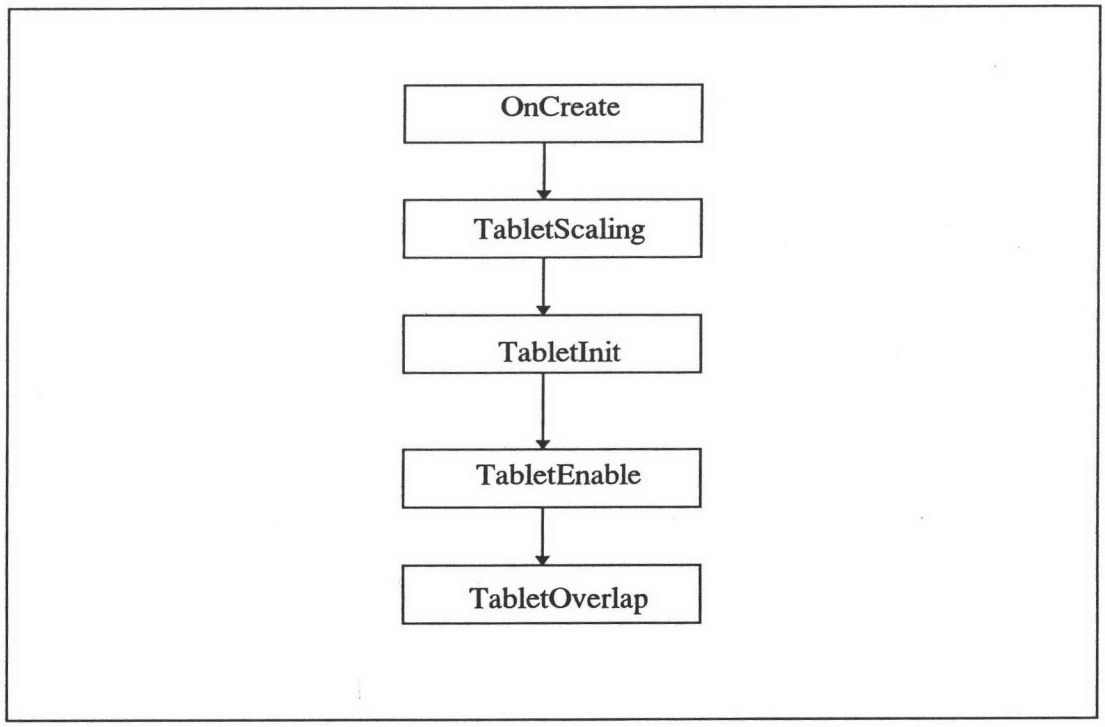
มาตรฐาน Wintab ได้กำหนดวิธีการในการติดต่อระหว่างโปรแกรมประยุกต์และเครื่องอ่านพิกัด โดยใช้ Wintab.DLL เป็นตัวให้บริการแก่โปรแกรมประยุกต์ โปรแกรมประยุกต์จะมองบริการของ Wintab เป็นวัตถุ โดยเรียกว่าคอนเท็กซ์ (Context) โปรแกรมประยุกต์สามารถกำหนดขอบเขตการรับข้อมูลและความละเอียดผ่านทางคอนเท็กซ์ และโปรแกรมประยุกต์หลายโปรแกรม สามารถทำงานติดต่อกับเครื่องอ่านพิกัดพร้อมกันโดยมีคอนเท็กซ์ที่ต่างกันได้ดังแสดงในรูป 5.20



รูปที่ 5.20 แสดงการติดต่อระหว่างโปรแกรมประยุกต์กับ Wintab.DLL

โปรแกรมการวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะ มีการติดต่อกับ Wintab.DLL โดยผ่านทางวัตถุ CDigView ด้วยฟังก์ชัน Oncreate()

ฟังก์ชัน Oncreate( ) ของวัตถุ CDigView จะทำการสร้างคอนเท็กซ์ และกำหนดค่าเริ่มต้นของเครื่องอ่านพิกัด โดยขั้นแรกจะทำการอ่านค่าอัตราความละเอียดด้วยฟังก์ชัน TabletScaling( ) เก็บไว้ในตัวแปร m_scale ต่อจากนั้นจะทำการสร้างคอนเท็กซ์ด้วยฟังก์ชัน TabletInit( ) สุดท้ายใช้ฟังก์ชัน TabletEnable( ) และ TabletOverlap( ) เพื่อกำหนดให้เครื่องอ่านพิกัดเริ่มการทำงานดังแสดงในรูปที่ 5.21



รูปที่ 5.21 แสดงขั้นตอนการติดต่อกับเครื่องอ่านพิกัด

และสามารถแสดงขั้นตอนการติดต่อกับเครื่องอ่านพิกัดเป็นภาษา C++ ได้ดังรูปที่ 5.22

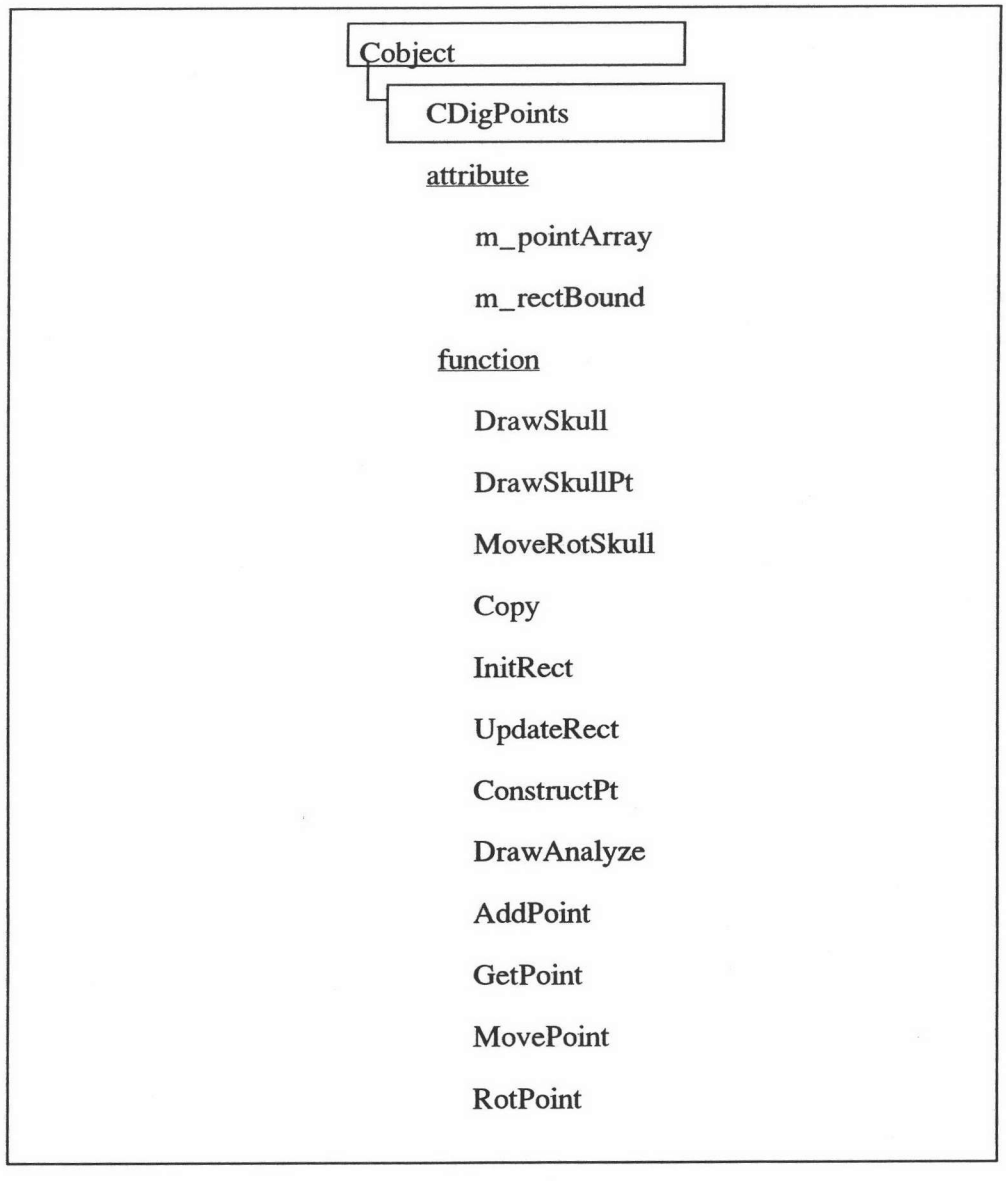
```
int CDigView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
if (CView::OnCreate(lpCreateStruct) == -1)
        return -1;
// Initial Tablet
TabletScaling(m_scale);
m_hTab=TabletInit(this->GetSafeHwnd(),m_scale);
TabletEnable(m_hTab,TRUE);
TabletOverlap(m_hTab,TRUE);
// Create New CDigPoint
m_pDigCur    = GetDocument()->NewDigitize();
m_bDig                = TRUE;
m_pDigCur->InitRect();
SetCapture();
return 0;
}
```

รูปที่ 5.22 แสดงแสดงขั้นตอนการติดต่อกับเครื่องอ่านพิกัดโดยใช้ภาษา C++

หลังจากทำการสร้างคอนเท็กซ์แล้ว โปรแกรมขับของเครื่องอ่านพิกัดจะส่งข้อความมายังโปรแกรมวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะ และวัตถุ CDigView จะได้รับข้อความดังกล่าว และส่งให้ฟังก์ชัน OnWTPacket( ) เพื่อทำการแสดงผลและบันทึกจุดเมื่อมีการกดปุ่มที่ตัวชี้ตำแหน่ง

2. ส่วนการสร้างภาพโครงร่างของกระโหลกศีรษะ

ในการสร้างภาพโครงร่างของกระโหลกศีรษะนั้น ผู้วิจัยได้กำหนดวัตถุอีกหนึ่งชนิด เพื่อใช้ในการเก็บจุดที่ได้จากเครื่องอ่านพิกัด โดยใช้ชื่อว่า CDigPoints และมีโครงสร้างข้อมูล ดังแสดงในรูปที่ 5.23



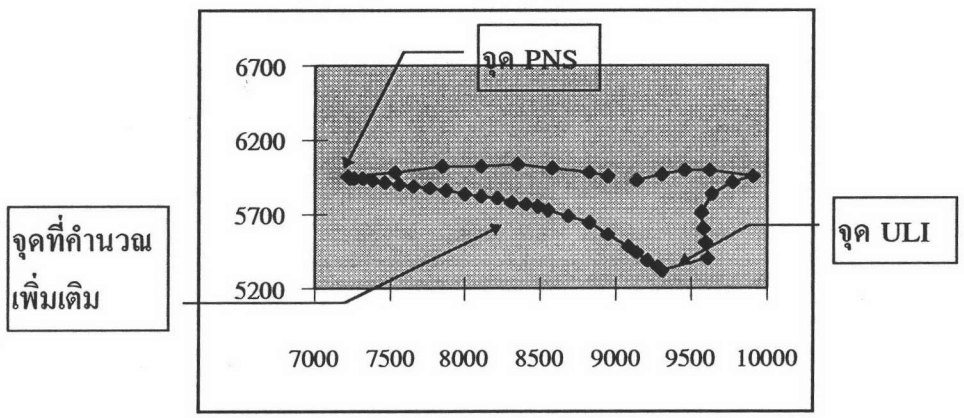
รูปที่ 5.23 แสดงคลาส CDigPoints

วัตถุ CDigPoints เป็นคลาสที่ได้รับการถ่ายทอดมาจากคลาส CObject ประกอบด้วยข้อมูล m_pointArray ซึ่งใช้ในการเก็บจุดที่ได้จากเครื่องอ่านพิกัด โดยเก็บในลักษณะเป็นแถวลำดับและ m_rectBound เป็นข้อมูลซึ่งบอกขอบเขตของรูปโครงร่างของกระโหลกศีรษะ

วัตถุ CDigPoints สามารถแสดงภาพโครงร่างกระโหลกศีรษะด้วยฟังก์ชัน DrawSkull( ) สามารถหมุนภาพโครงร่างกระโหลกศีรษะด้วยฟังก์ชัน MoveRotSkull( ) และสามารถเปลี่ยนแปลงเฉพาะบางจุดได้ด้วยฟังก์ชัน MovePoint( ), RotPoint( )

ในการสร้างภาพโครงร่างกระโหลกศีรษะนั้นจะสร้างภาพโดยอาศัยความสัมพันธ์ระหว่างจุดสองจุดที่อยู่ใกล้กัน และทำการคำนวณหาจุดเพิ่มเติมซึ่งอยู่ระหว่างจุดทั้งสองจากสมการที่ได้กำหนดไว้ล่วงหน้า ต่อจากนั้นจะทำการลากเส้นเพื่อเชื่อมต่อจุดทั้งหมดเข้าด้วยกัน ด้วยหลักการเดียวกันจะทำการสร้างเส้นระหว่างจุดทั้งหมดที่ได้ทำการอ่านจากเครื่องอ่านพิกัดเพื่อแสดงเป็นภาพโครงร่างของกระโหลกศีรษะที่สมบูรณ์

ตัวอย่างการสร้างเส้นจากจุด ULI ไปยังจุด PNS ของกระดูกขากรรไกรบน แสดงในรูปที่ 5.24 ผู้วิจัยได้ใช้สมการในรูปที่ 5.25 ในการคำนวณหาจุดระหว่างจุด ULI และจุด PNS และเมื่อได้จุดตามที่ต้องการแล้วจึงทำการลากเส้นเชื่อมจุดทั้งหมดเข้าด้วยกัน



รูปที่ 5.24 แสดงภาพการสร้างภาพโครงร่างของขากรรไกรบน

XULi	YULi
XULi-(28/2088)*(XULi-XPNS)	YULi+(18/628)*(YPNS-YULi)
XULi-(93/2088)*(XULi-XPNS)	YULi+(63/628)*(YPNS-YULi)
XULi-(169/2088)*(XULi-XPNS)	YULi+(114/628)*(YPNS-YULi)
XULi-(227/2088)*(XULi-XPNS)	YULi+(154/628)*(YPNS-YULi)
XULi-(353/2088)*(XULi-XPNS)	YULi+(241/628)*(YPNS-YULi)
XULi-(480/2088)*(XULi-XPNS)	YULi+(321/628)*(YPNS-YULi)
XULi-(481/2088)*(XULi-XPNS)	YULi+(321/628)*(YPNS-YULi)
XULi-(614/2088)*(XULi-XPNS)	YULi+(367/628)*(YPNS-YULi)
XULi-(749/2088)*(XULi-XPNS)	YULi+(407/628)*(YPNS-YULi)
XULi-(819/2088)*(XULi-XPNS)	YULi+(425/628)*(YPNS-YULi)
XULi-(902/2088)*(XULi-XPNS)	YULi+(443/628)*(YPNS-YULi)
XULi-(996/2088)*(XULi-XPNS)	YULi+(461/628)*(YPNS-YULi)
XULi-(1097/2088)*(XULi-XPNS)	YULi+(481/628)*(YPNS-YULi)
XULi-(1206/2088)*(XULi-XPNS)	YULi+(499/628)*(YPNS-YULi)
XULi-(1317/2088)*(XULi-XPNS)	YULi+(517/628)*(YPNS-YULi)
XULi-(1431/2088)*(XULi-XPNS)	YULi+(535/628)*(YPNS-YULi)
XULi-(1541/2088)*(XULi-XPNS)	YULi+(553/628)*(YPNS-YULi)
XULi-(1650/2088)*(XULi-XPNS)	YULi+(567/628)*(YPNS-YULi)
XULi-(1751/2088)*(XULi-XPNS)	YULi+(582/628)*(YPNS-YULi)
XULi-(1844/2088)*(XULi-XPNS)	YULi+(594/628)*(YPNS-YULi)
XULi-(1925/2088)*(XULi-XPNS)	YULi+(606/628)*(YPNS-YULi)
XULi-(1994/2088)*(XULi-XPNS)	YULi+(614/628)*(YPNS-YULi)
XULi-(2045/2088)*(XULi-XPNS)	YULi+(622/628)*(YPNS-YULi)
XULi-(2078/2088)*(XULi-XPNS)	YULi+(626/628)*(YPNS-YULi)
XPNS	YPNS

รูปที่ 5.25 แสดงสมการที่ใช้ในการคำนวณระหว่างจุด ULI และ PNS

และสามารถแสดงโครงสร้างข้อมูลของวัตถุ CDigPoints ด้วยภาษา C++ ดังแสดงในรูปที่ 5.26

```
class CDigPoints : public CObject
{
public:
        CDigPoints();
protected:
        //CDigPoints();
        DECLARE_SERIAL(CDigPoints)
// Attributes
        CDWordArray         m_pointArray;
        CRect               m_rectBound;
//operations
public:
        BOOL DrawSkull(CDC* pDC);
        BOOL DrawSkullPt(CDC* pDC,int nPoint,BOOL bUpdateRect);
        void MoveRotSkull(int nX,int nY,double dRadian,CPoint ptOrg);
        CDigPoints* Copy();
        void InitRect(){m_rectBound.SetRectEmpty();}
        void UpdateRect(POINT* pPoints,int nNumPoint);
        void ConstructPt();
        BOOL DrawAnalyze(CDC* pDC,UINT nAnaType);
        void AddPoint(CPoint pt);
        CPoint GetPoint(int i) const { return CPoint(m_pointArray[i]); }
        CPoint* RtvPoint(int nI){ return (CPoint*)(&m_pointArray[nI]);}
        void MovePoint(CPoint* pPtOld,int nX,int nY);
        void RotPoint(CPoint* pPtOld,double dRadian,CPoint ptOrg);
}
```

รูปที่ 5.26 แสดงโครงสร้างข้อมูลของ CDigPoints ด้วยภาษา C+

Foley and Dam (1982) ได้อธิบายถึงวิธีการเปลี่ยนแปลงจุดในระนาบ 2 มิติ ว่าจุดใดๆในระนาบสามารถถูกเคลื่อนย้ายได้โดยการบวกค่าที่ต้องการเปลี่ยนแปลงให้กับจุดนั้น ตัวอย่างเช่นจุด P(x,y) สามารถถูกเคลื่อนไปในแนวแกน X เป็นระยะทาง Dx และถูกเคลื่อนไปในแนวแกน Y เป็นระยะทาง Dy ไปยังจุด P′(x′,y′) ได้ โดยสมการดังนี้

$$x' = x + Dx$$

และ

$$y' = x + Dy$$

นอกจากนี้จุดใดๆ ในระนาบ 2 มิติ สามารถถูกหมุนรอบจุดกำเนิดได้ โดยสมการดังนี้

$$x' = x \cdot \cos(\theta) - y \cdot \sin(\theta)$$

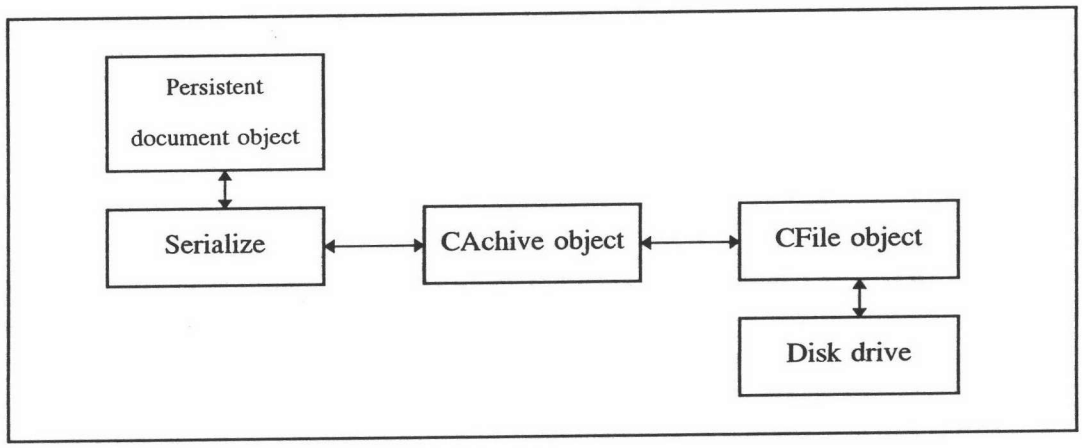
และ

$$y' = x \cdot \sin(\theta) + y \cdot \cos(\theta)$$

จากหลักการดังกล่าวสามารถใช้ในการเคลื่อนย้าย หรือหมุน วัตถุ CDigPoints ได้โดยการเปลี่ยนแปลงจุดที่เก็บอยู่ใน m_pointArray โดยถ้าทำการเปลี่ยนแปลงทุกจุดพร้อมกันด้วยสมการเดียวกันจะเป็นการเคลื่อนหรือหมุน ภาพโครงร่างของกระโหลกศีรษะ แต่ถ้าทำการเปลี่ยนแปลงเฉพาะบางจุดหรือทำการเปลี่ยนแปลงด้วยสมการที่ต่างกันในแต่ละจุดก็จะเป็นการเปลี่ยนแปลงลักษณะของภาพโครงร่างของกระโหลกศีรษะ

3. ส่วนการเก็บรักษาและค้นคืนข้อมูล

Kruglinski (1993) ได้อธิบายว่าการเขียนโปรแกรม โดยใช้โครงสร้างมาตรฐาน (Application Framework) นั้น เมื่อผู้ใช้เลือกเมนูเพื่อเก็บรักษาหรือเรียกคืนข้อมูล ส่วนเอกสารของโครงสร้างมาตรฐานจะทำการเรียกฟังก์ชัน Serialize( ) โดยอัตโนมัติ ฟังก์ชัน Serialize( ) จะรับหรือส่งข้อมูลผ่านวัตถุ CArchive โดยวัตถุ CArchive จะรับและส่งผ่านข้อมูลให้กับวัตถุ CFile เพื่อจัดเก็บหรือค้นคืนข้อมูล ดังแสดงในรูปที่ 5.27



รูปที่ 5.27 แสดงการเก็บรักษาและค้นคืนข้อมูลผ่านฟังก์ชัน Serialize( )

ในการวิจัยนี้ โปรแกรมวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะ ใช้วัตถุ CCepdoc ในการเก็บข้อมูลของโปรแกรม และมีฟังก์ชัน Serialize( ) เพื่อใช้ในการเก็บรักษาหรือค้นคืนข้อมูล ดังแสดงในรูปที่ 5.28

```
void CCepDoc::Serialize(CArchive& ar)
{
        if (ar.IsStoring())
        {
                // TODO: add storing code here   serialize
                ar << m_sName;
```

รูปที่ 5.28 แสดงฟังก์ชัน Serialize( ) ของวัตถุ CCepdoc

```
            ar << m_sBirth;
            ar << m_sRace;
            ar << (WORD)m_nSex;
        }
        else
        {
            // TODO: add loading code here
            ar >> m_sName;
            ar >> m_sBirth;
            ar >> m_sRace;
            WORD w;
            ar >> w;
            m_nSex =(int)w;
        }
        m_infRecList.Serialize(ar);
}
```

รูปที่ 5.28 แสดงฟังก์ชัน Serialize( ) ของวัตถุ Ccepdoc (ต่อ)

4. ส่วนการวิเคราะห์ผล

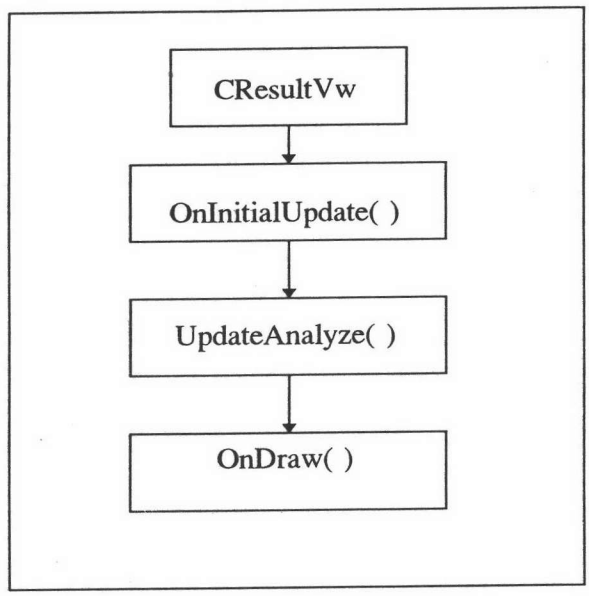
ในการแสดงผลการวิเคราะห์ภาพถ่ายรังสีเอกซ์ของกระโหลกศีรษะนั้น จะทำการแสดงผลผ่านมุมมอง CResultVw โดย CResultVw เป็นคลาสที่ถ่ายทอดมาจากคลาส CScrollView ทำให้วินโดว์ที่สร้างขึ้นสำหรับมุมมองที่สามารถเลื่อนข้อมูลขึ้นลงเพื่อดูได้

ขั้นตอนการทำงานของวัตถุ CResultVw คือเมื่อผู้ใช้เลือกเมนู Window-Result ข้อความจะถูกส่งผ่านไปยังวัตถุ CMainFrame และทำการสร้างมุมมอง CResultVw ด้วยฟังก์ชัน OnWindowResult( ) โดยมีรายละเอียดของฟังก์ชัน ดังแสดงในรูปที่ 5.29

```
void CMainFrame::OnWindowResult()
{
        CMDIChildWnd* pActiveChild = MDIGetActive();
    CDocument* pDocument;
    if (pActiveChild == NULL ||
      (pDocument = pActiveChild->GetActiveDocument()) == NULL)
    {
      TRACE0("Warning: No active document for WindowNew
                command\n");
       AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
      return;     // command failed
    }
   // otherwise we have a new frame !
    CDocTemplate* pTemplate = ((CCepApp*) AfxGetApp())->
                    m_pTemplate2;
    ASSERT_VALID(pTemplate);
    CFrameWnd* pFrame = pTemplate->CreateNewFrame(pDocument,
pActiveChild);
   if (pFrame == NULL)
    {
      TRACE0("Warning: failed to create new frame\n");
      AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
     return;     // command failed
   }
   pTemplate->InitialUpdateFrame(pFrame, pDocument);
}
```

รูปที่ 5.29 แสดงฟังก์ชัน OnWindowResult( ) ของ CMainFrame

เมื่อมุมมอง CResultVw ถูกสร้างขึ้น CResultVw จะเรียกฟังก์ชัน OnInitUpdate( ) โดยอัตโนมัติ ฟังก์ชัน OnInitUpdate( ) จะเรียกฟังก์ชัน UpdateAnalyze( ) เพื่อทำการคำนวณผลการวิเคราะห์เก็บไว้ในตัวแปร m_StringArray และฟังก์ชัน OnDraw( ) แสดงจะผลการวิเคราะห์บนหน้าจอ ดังแสดงในรูปที่ 5.30



รูปที่ 5.30 แสดงขั้นตอนการทำงานของมุมมอง CResultVw

5. ส่วนการบันทึกหมายเหตุ

ในส่วนการบันทึกหมายเหตุจะทำการติดต่อผ่านทางมุมมอง CNoteVw โดย CNoteVw เป็นคลาสที่ถ่ายทอดมาจากคลาส CEditView ซึ่งมีคุณสมบัติพื้นฐานของโปรแกรมบรรณาธิกรณ์อยู่ด้วย แต่ก็มีข้อจำกัดอยู่มาก เช่น ไม่สามารถแสดงแบบอักษรได้มากกว่าครั้งละ 1 แบบ และเก็บข้อมูลตัวอักษรได้สูงสุด 42 KB

ขั้นตอนการทำงานของวัตถุ CNoteVw จะเริ่มจากเมื่อผู้ใช้เลือกเมนู Window-Note ข้อความจะถูกส่งผ่านไปยังวัตถุ CMainframe และทำการสร้างมุมมอง CNoteVw ด้วยฟังก์ชัน OnWindowNote( ) โดยมีรายละเอียดของฟังก์ชัน แสดงในรูปที่ 5.31

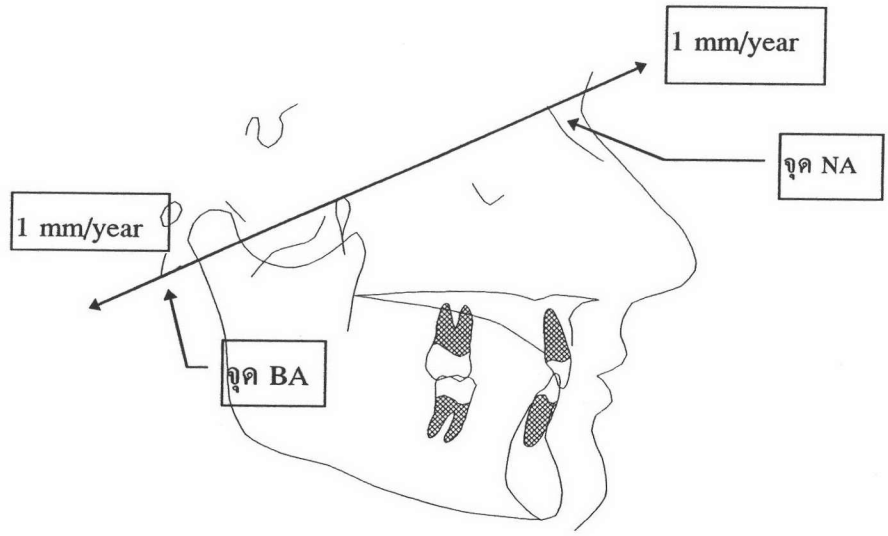
```
void CMainFrame::OnWindowNote()
{
        CMDIChildWnd* pActiveChild = MDIGetActive();
    CDocument* pDocument;
    if (pActiveChild == NULL ||
      (pDocument = pActiveChild->GetActiveDocument()) == NULL)
    {
      TRACE0("Warning: No active document for WindowNew command\n");
      AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
      return;     // command failed
    }
  // otherwise we have a new frame !
    CDocTemplate* pTemplate = ((CCepApp*) AfxGetApp())->
                                    m_pTemplate4;
    ASSERT_VALID(pTemplate);
    CFrameWnd* pFrame = pTemplate->CreateNewFrame(pDocument,
pActiveChild);
    if (pFrame == NULL)
    {
      TRACE0("Warning: failed to create new frame\n");
      AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
      return;     // command failed
    }
  pTemplate->InitialUpdateFrame(pFrame, pDocument);
}
```

รูปที่ 5.31 แสดงฟังก์ชัน OnWindowNote( ) ของ CMainFrame

6. ส่วนการทำนายการเปลี่ยนแปลงของกระดูกบริเวณใบหน้าและขากรรไกรที่เกิดจากการเจริญเติบโต

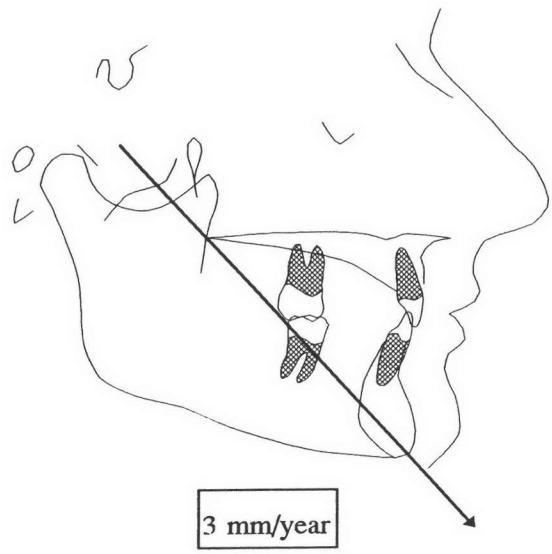
Rickettes, Roth, Chaconas, Schulhof (1982) ได้อธิบายถึงหลักการทำนายการเจริญเติบโตของกระโหลกศีรษะดังนี้

กระดูกส่วนฐานของกระโหลกศีรษะจะมีการเจริญเติบโตไปตามแนวเส้นที่ลากระหว่างจุด NA และ BA ด้านละ 1 มิลลิเมตรต่อปี ดังแสดงในรูปที่ 5.32



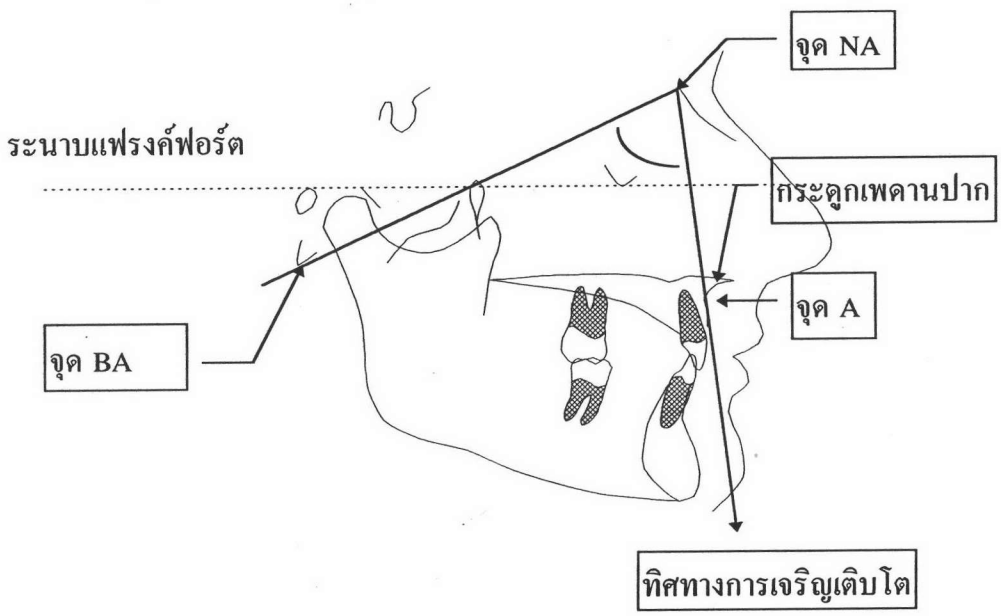
รูปที่ 5.32 แสดงการเจริญเติบโตของฐานกระโหลกศีรษะ

กระดูกส่วนคางจะมีการเจริญเติบโตตามแนวแกนใบหน้า (Facial Axis) 3 มิลลิเมตรต่อปี ดังแสดงในรูปที่ 5.33



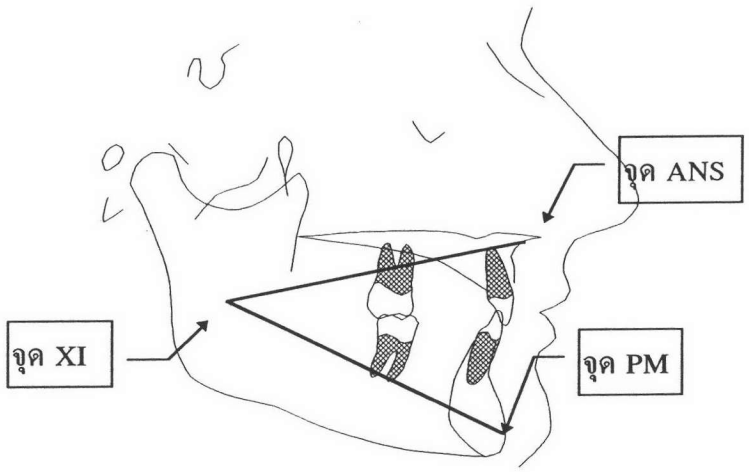
รูปที่ 5.33 แสดงการเจริญเติบโตของกระดูกส่วนคาง

กระดูกบริเวณเพดานปากจะเจริญเติบโตขนานกับระนาบแฟรงค์ฟอร์ต และมุมระหว่างเส้น BA-NA และ NA-A ต้องคงที่ ดังแสดงในรูปที่ 5.34



รูปที่ 5.34 แสดงการเจริญเติบโตของกระดูกบริเวณเพดานปาก

นอกจากนี้มุม ระหว่างเส้น ANS-XI และ XI-PM ต้องคงที่อยู่เสมอ ดังแสดงในรูปที่ 5.35



รูปที่ 5.35 แสดงการเจริญเติบโตโดยมุม ANS-XI-PM คงที่

จากทฤษฎีการทำนายการเจริญเติบโตดังกล่าว ผู้วิจัยได้ทำการเปลี่ยนแปลงตำแหน่งของจุดตามทฤษฎี ในวัตถุ CDigPoints โดยใช้ฟังก์ชัน MovePoint เพื่อทำการเปลี่ยนแปลงตำแหน่งของจุดทีละจุด เมื่อทำการเปลี่ยนแปลงจนครบทุกจุดแล้ว จึงสั่งให้วัตถุ CDigPoints ทำการแสดงภาพโครงร่างของกระโหลกศีรษะด้วย ฟังก์ชัน DrawSkull ก็จะได้ภาพโครงร่างของกระโหลกศีรษะหลังการเจริญเติบโต

ตัวอย่างเช่นการเจริญเติบโตของจุด NA และ BA ดังแสดงในรูปที่ 5.32 จุด NA และ BA จะเจริญเติบโตไปด้านละ 1 มิลลิเมตรตามแนวแกน NA-BA ในขั้นตอนการเคลื่อนจุด NA และ BA นั้นผู้วิจัยจะทำการคำนวณหามุมของแนวแกน NA-BA ที่ทำกับระนาบแกน X โดยใช้สมการดังนี้

$\theta = \arctan((y1-y2) / (x1-x2))$

เมื่อ θ คือมุมที่ทำกับระนาบแกน X

NA คือจุด (x1,y1)

และ BA คือจุด (x2,y2)

ต่อจากนั้นจะหาค่าการเพิ่มในแนวแกน X และ Yโดยใช้สมการ

$Dx = D . \cos(\theta)$

$Dy = D . \sin(\theta)$

เมื่อ D คือระยะทางที่เจริญเติบโตเพิ่มขึ้นในแนวแกน NA-BA

Dx คือระยะทางการเพิ่มในแนวแกน X

Dy คือระยะทางการเพิ่มในแนวแกน Y

และ θ คือมุมที่ทำกับระนาบแกน X

จากนั้นจะทำการเปลี่ยนแปลงจุด NA โดยการบวกค่า Dx และ Dy เข้ากับจุด NA และทำการเปลี่ยนแปลงจุด BA โดยการลบค่า Dx และ Dy ออกจาก จุด BA

ในส่วนการทำนายการเปลี่ยนแปลงของกระดูกบริเวณใบหน้า และขากรรไกรที่เกิดจากการเจริญเติบโต จะทำการติดต่อกับผู้ใช้ผ่านทางมุมมอง CGrowthVw โดย CGrowthVw เป็นคลาสที่ถ่ายทอดมาจากคลาสCView

ขั้นตอนการทำงานของวัตถุ CGrowthVw จะเริ่มจากเมื่อผู้ใช้เลือกเมนู Trace-Growth Forecast ข้อความจะถูกส่งผ่านไปยังวัตถุ CMainframe และทำการสร้างมุมมอง CGrowthVw ด้วยฟังก์ชัน OnTraceGrowthForecast() โดยมีรายละเอียดของฟังก์ชันดังแสดงในรูปที่ 5.36

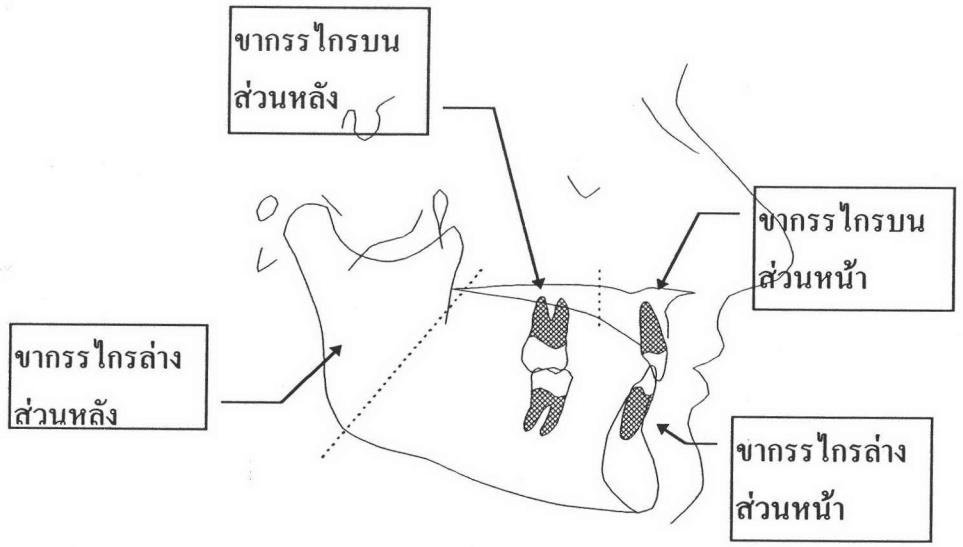
```
Void CMainFrame::OnTraceGrowthforcast()
{
        CMDIChildWnd* pActiveChild = MDIGetActive();
    CDocument* pDocument;
    if (pActiveChild == NULL ||
       (pDocument = pActiveChild->GetActiveDocument()) == NULL)
    {
     TRACE0("Warning: No active document for WindowNew command\n");
     AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
     return;     // command failed
    }
   // otherwise we have a new frame !
   CDocTemplate* pTemplate=((CCepApp*) AfxGetApp())->m_pTemplate5;
   ASSERT_VALID(pTemplate);
   CFrameWnd* pFrame = pTemplate->CreateNewFrame(pDocument,
                                          pActiveChild);
   if (pFrame == NULL)
    {
        TRACE0("Warning: failed to create new frame\n");
        AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
        return;     // command failed
     }
    pTemplate->InitialUpdateFrame(pFrame, pDocument);
}
```

รูปที่ 5.36 แสดงฟังก์ชัน OnTraceGrowthForecast( ) ของ Cmainframe

7. ส่วนการจำลองการเปลี่ยนแปลงของกระดูกบริเวณใบหน้าและขากรรไกรที่เกิดขึ้นเนื่องจากการผ่าตัด

ในส่วนการจำลองการเปลี่ยนแปลงของกระดูกบริเวณใบหน้า และขากรรไกรที่เกิดจากการผ่าตัด จะมีลักษณะการทำงานคล้ายกับการทำนายการเปลี่ยนแปลงเนื่องจากการเจริญเติบโต ต่างกันที่การผ่าตัดจะเป็นการเปลี่ยนแปลงเฉพาะตำแหน่งไม่ได้เปลี่ยนแปลงทั้งกระโหลกศีรษะ และตำแหน่งที่สามารถเปลี่ยนแปลงได้มีจำกัด

ตำแหน่งที่สามารถเปลี่ยนแปลงได้จากการผ่าตัดมี 6 ส่วนคือ ขากรรไกรบนส่วนหน้า ขากรรไกรบนส่วนหลัง ขากรรไกรล่างส่วนหน้า ขากรรไกรล่างส่วนหลัง ขากรรไกรบนทั้งหมด และขากรรไกรล่างทั้งหมด ดังแสดงในรูปที่ 5.37



รูปที่ 5.37 แสดงชิ้นส่วนของกระดูกที่สามารถทำการผ่าตัด

ในการจำลองการผ่าตัด โปรแกรมจะทำการเปลี่ยนแปลงข้อมูลของจุดในชิ้นกระดูกที่ถูกเลือก โดยใช้ฟังก์ชัน MovePoint ของวัตถุ CDigPoints ต่อจากนั้นจะทำการเปลี่ยนแปลงจุดบริเวณใบหน้าเพื่อให้รักษาความสัมพันธ์ระหว่างกระดูกและเนื้อเยื่ออ่อนไว้ สุดท้ายจะใช้ฟังก์ชัน DrawSkull เพื่อแสดงรูปโครงร่างกระโหลกศีรษะหลังจากการผ่าตัด

ในส่วนการจำลองการเปลี่ยนแปลงของกระดูกบริเวณใบหน้า และขากรรไกรที่เกิดจากการผ่าตัด จะทำการติดต่อกับผู้ใช้ผ่านทางมุมมอง CTreatVw โดย CTreatVwเป็นคลาสที่ถ่ายทอดมาจากคลาสCView

ขั้นตอนการทำงานของวัตถุ CTreatVw จะเริ่มจากเมื่อผู้ใช้เลือกเมนู Trace-Treatment Plan ข้อความจะถูกส่งผ่านไปยังวัตถุ CMainframe และทำการสร้างมุมมอง CTreatVw ด้วยฟังก์ชัน OnTraceTreatmentPlan() โดยมีรายละเอียดของฟังก์ชัน ดังแสดงในรูปที่ 5.38

```
void CMainFrame::OnTraceTreatmentplan()
{
        CMDIChildWnd* pActiveChild = MDIGetActive();
    CDocument* pDocument;
 if (pActiveChild == NULL ||

   (pDocument = pActiveChild->GetActiveDocument()) == NULL)
  {
      TRACE0("Warning: No active document for WindowNew
              command\n");
      AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
      return;     // command failed
   }
```

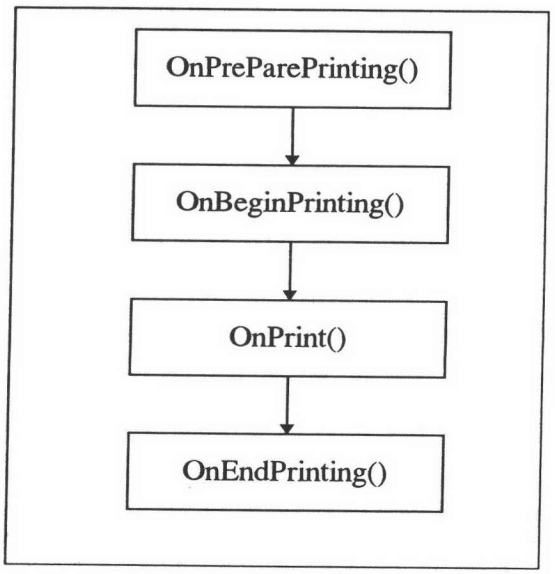
รูปที่ 5.38 แสดงฟังก์ชัน OnTraceTreatmentPlan() ของ Cmainframe

```
    // otherwise we have a new frame !
     CDocTemplate* pTemplate = ((CCepApp*) AfxGetApp())->
                                        m_pTemplate6;
     ASSERT_VALID(pTemplate);
     CFrameWnd* pFrame = pTemplate->CreateNewFrame(pDocument,
                                        pActiveChild);
    if (pFrame == NULL)
    {
        TRACE0("Warning: failed to create new frame\n");
        AfxMessageBox(AFX_IDP_COMMAND_FAILURE);
     return;      // command failed
    }
    pTemplate->InitialUpdateFrame(pFrame, pDocument);
}
```

รูปที่ 5.38 แสดงฟังก์ชัน OnTraceTreatmentPlan() ของ Cmainframe (ต่อ)

7. ส่วนการพิมพ์

Kruglinski (1993) ได้อธิบายถึงการจัดการการพิมพ์ดังนี้ คลาสที่ได้รับการถ่ายทอดจากคลาส Cview จะทำการแสดงผลผ่านทางฟังก์ชัน OnDraw() โดยฟังก์ชัน OnDraw( ) อาจถูกเรียกโดยฟังก์ชัน OnPaint( ) เมื่อได้รับข้อความ WM_PAINT หรือถูกเรียกโดยฟังก์ชัน OnPrint( ) เมื่อผู้ใช้เลือกเมนูคำสั่งพิมพ์ นอกจากนี้ยังมีฟังก์ชันที่เกี่ยวข้องกับการพิมพ์อีกดังนี้ ฟังก์ชัน OnPreparePrinting ใช้ในการกำหนดจำนวนหน้า ฟังก์ชัน OnBeginPrinting ใช้ในการสร้างวัตถุ GDI เพื่อใช้ในการพิมพ์ ฟังก์ชัน EndPrinting ใช้ในการลบวัตถุ GDI เมื่อการพิมพ์สิ้นสุด และฟังก์ชันเหล่านี้มีความสัมพันธ์กัน ดังแสดงในรูปที่ 5.39



รูปที่ 5.39 แสดงความสัมพันธ์ของฟังก์ชันที่ใช้ในการพิมพ์

ในการวิจัยนี้วัตถุที่เป็นมุมมองจะมีฟังก์ชันในการพิมพ์ของตัวเองแยกเป็นอิสระ ทำให้แต่ละมุมมองสามารถกำหนดและจัดรูปแบบในการพิมพ์ของตนเองได้ ตัวอย่างเช่น ฟังก์ชันการพิมพ์ของคลาส CResultVw ดังแสดงในรูปที่ 5.40

```
// CResultVw printing
BOOL CResultVw::OnPreparePrinting(CPrintInfo* pInfo)
{
        // default preparation
        pInfo->SetMaxPage(1);
        return DoPreparePrinting(pInfo);
}
void CResultVw::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: add extra initialization before printing
}
void CResultVw::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
        // TODO: add cleanup after printing
}
void CResultVw::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
        pDC->SetMapMode(MM_TWIPS);
        OnDraw(pDC);
}
```

รูปที่ 5.40 แสดงฟังก์ชันการพิมพ์ของคลาส CResultVw