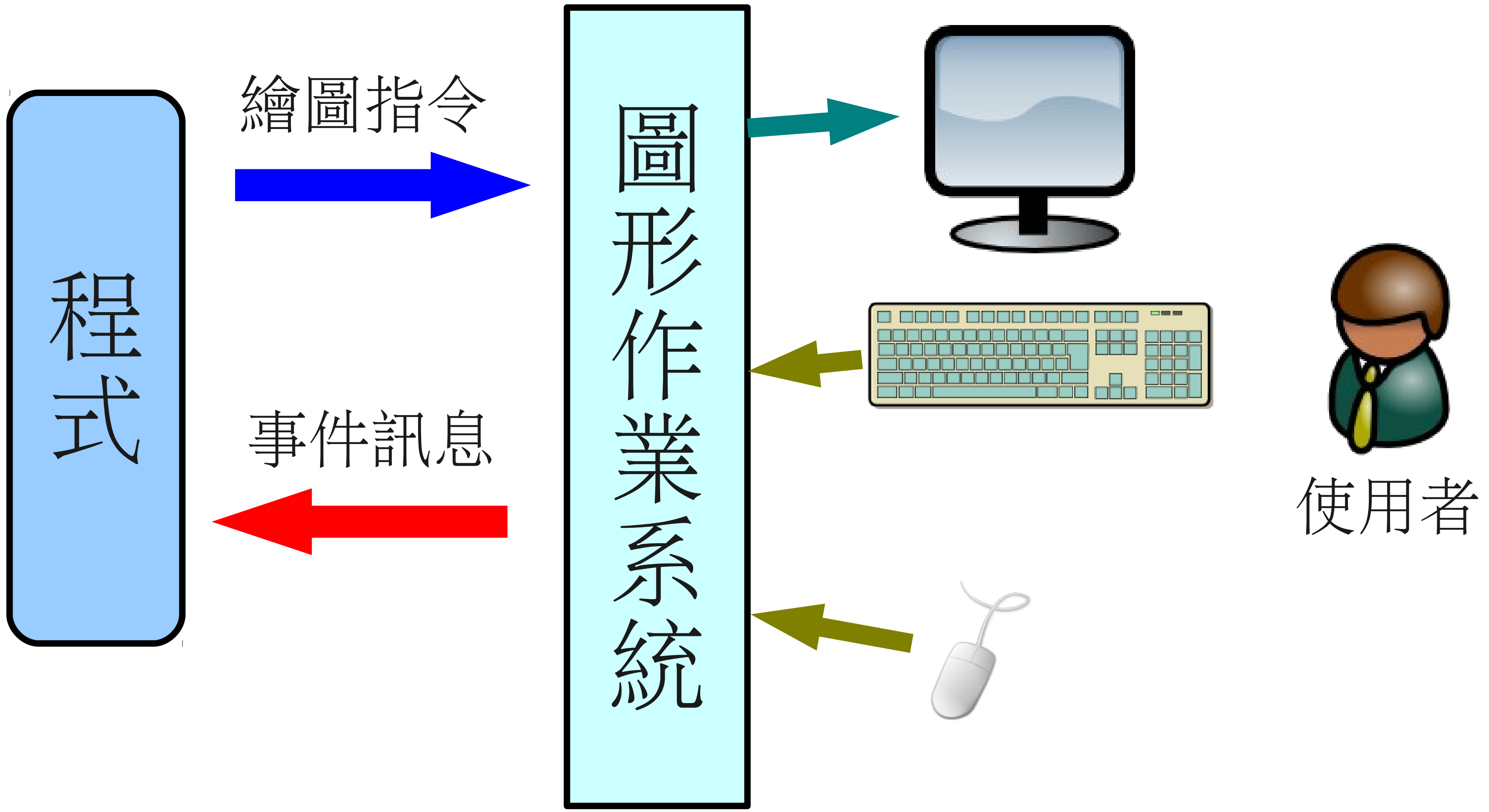
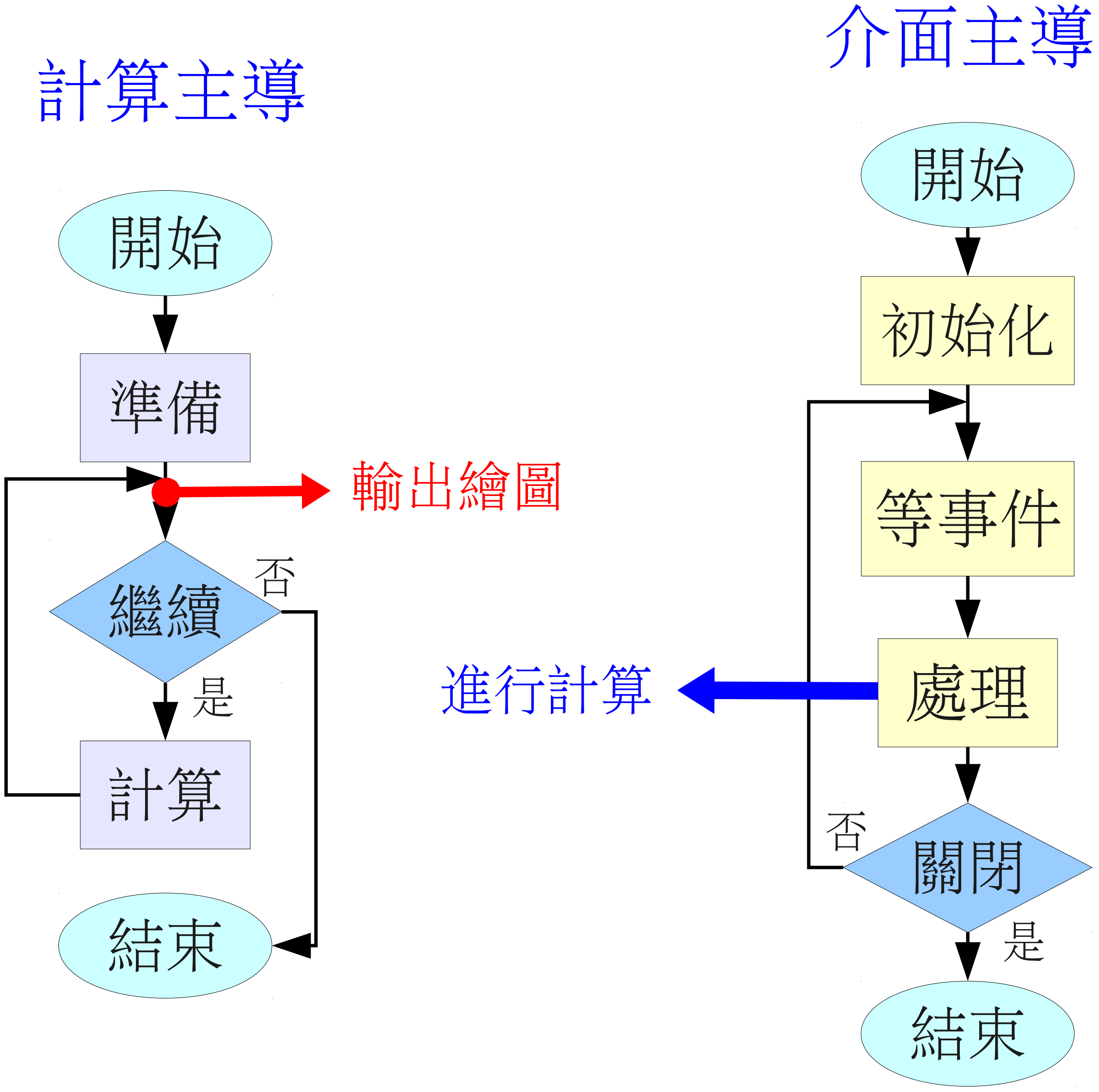
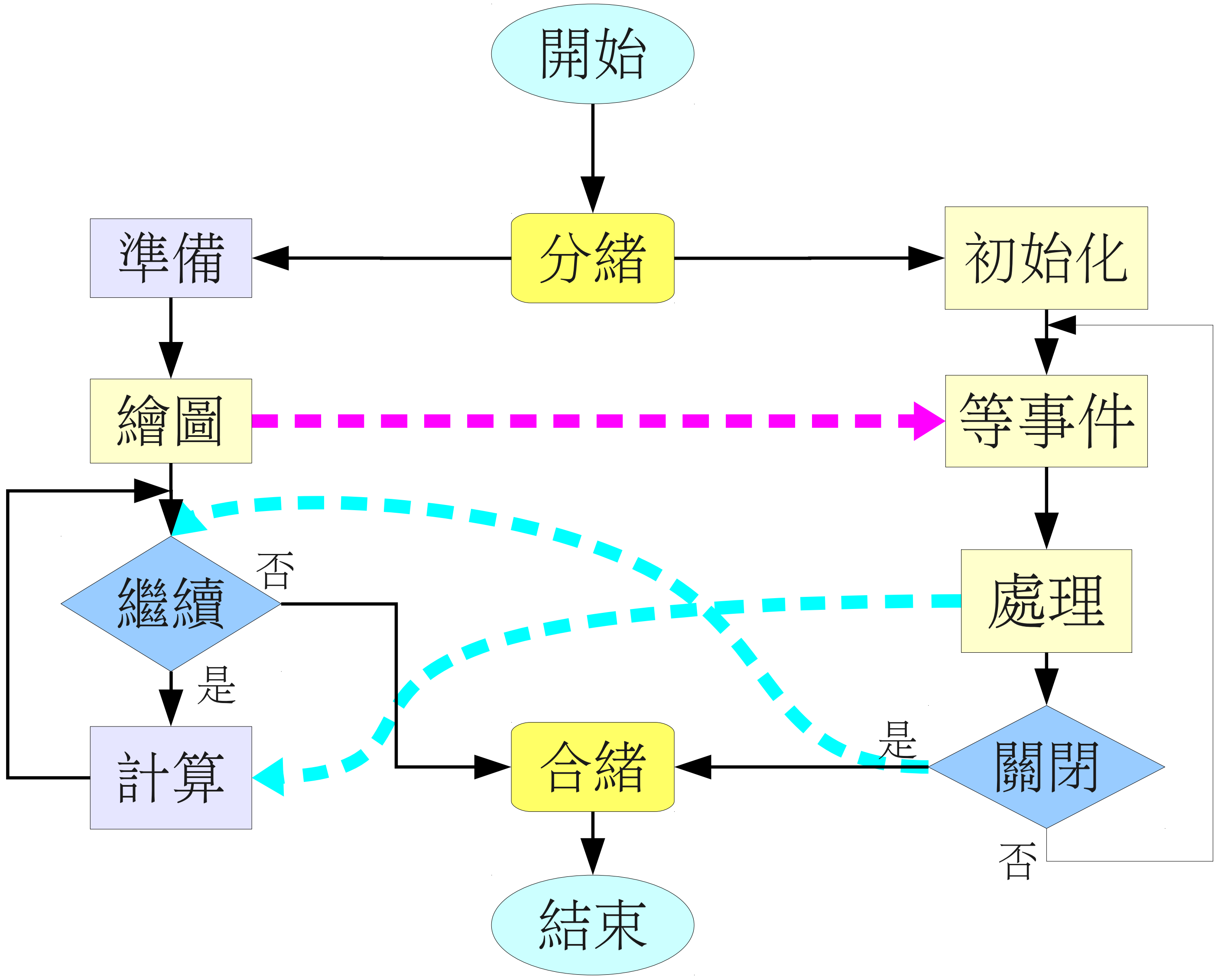
圖形介面系統



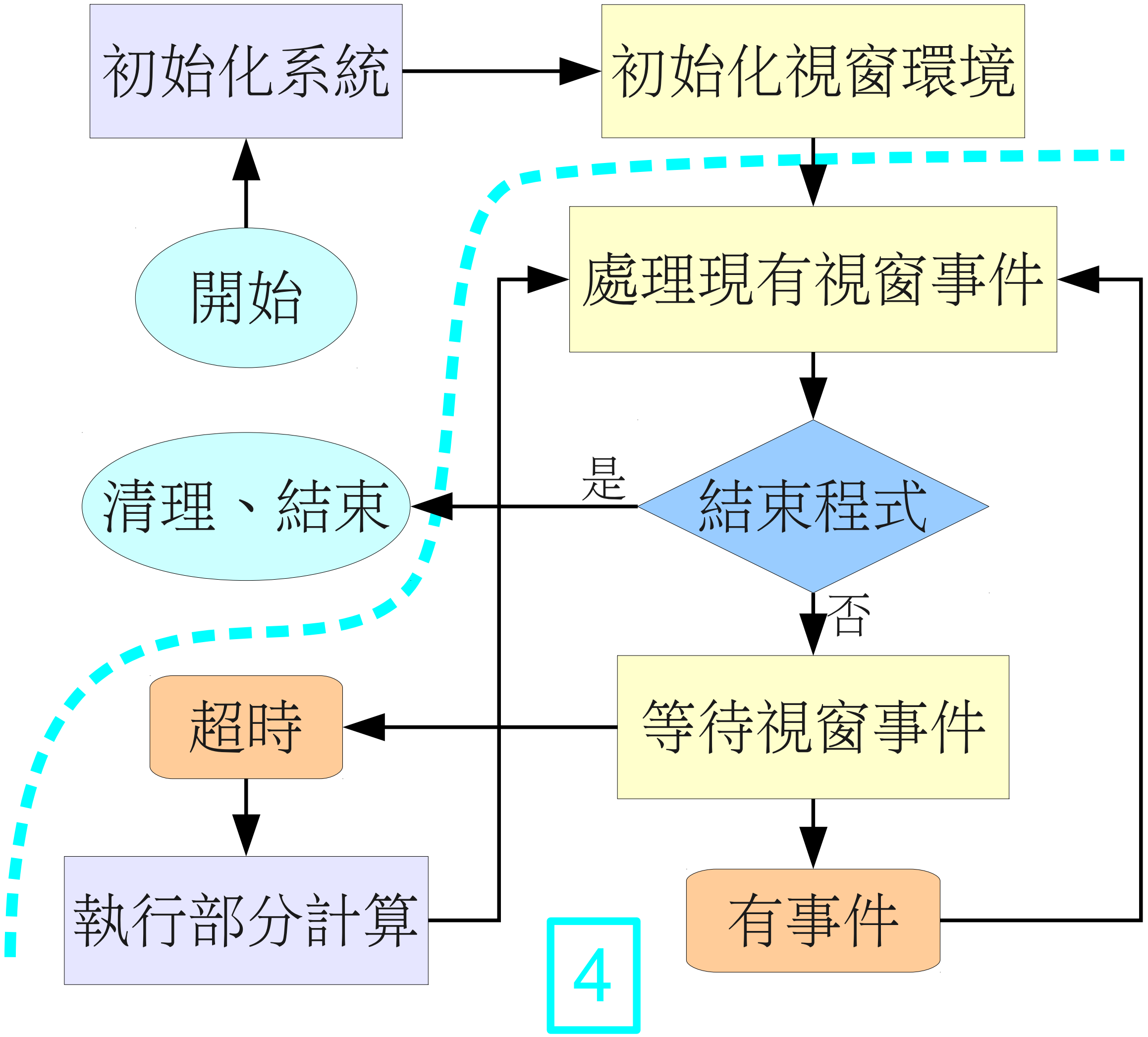
圖形輸出計算的主導權



多行緒 (multi-thread) 圖形程式

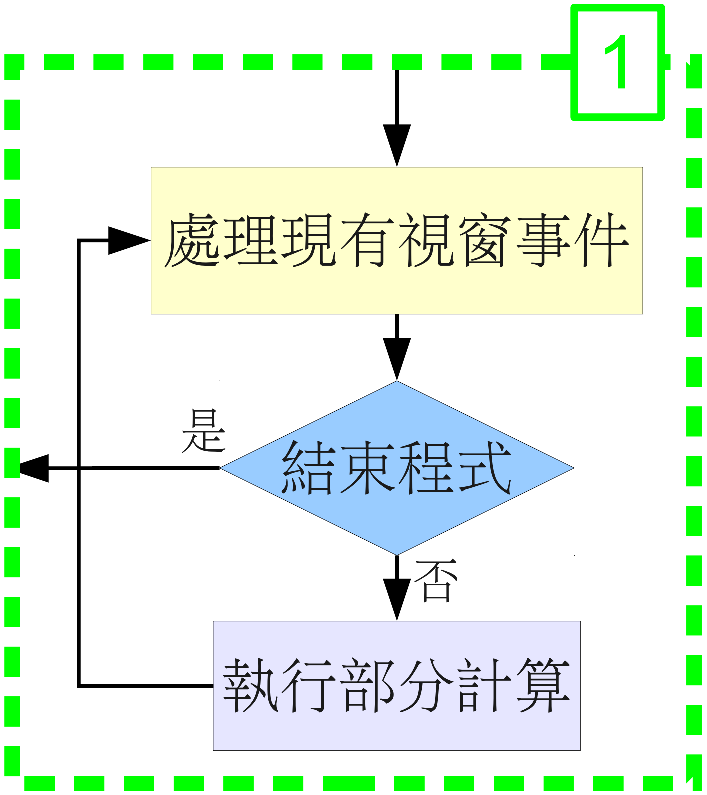


單行緒 (thread) 計算程式輸出



不同的計算輸出類型

1. 連續計算，　連續輸出
2. 連續計算，　定時更新
3. 節控步驟，　逐步輸出
4. 定時步驟，　逐步輸出



繪圖系統的一些概念

- Server Client Model
- Windows and Widgets
- Events and Messages
- Graphic Drawing Context

hk_step.cc 1/2

```
#include "draw_sys.hh"
#include "ran_nr.hh"
#include <iostream>
#include <iomanip>
#include <sstream>
class HKStep :
        public DrawSys
{
        RanNR rng;
        // system state
        size_t sz0, sz1, szz;
        bool * sites;
        // labeling
        int * lab, * nlab, lcnt;
        size_t scan;
        unsigned long * lclr;
        int find_lab(size_t i)
        {
                int l = lab[i];
                while (nlab[l] <= 0) l = - nlab[l];
                return l;
        }
        void draw(size_t i)
        {
                std::ostringstream os;
                size_t r = i / sz1, c = i % sz1;
                if (sites[i]) {
                        if (lab[i] >= 0) {
                                int l = find_lab(i);
                                set_color(lclr[l]);
                                draw_box(1 + c * 20, 1 + r * 20, 18, 18);
                                os.str(""); os << l;
                                set_color(0x000000);
                                draw_text(10 + c * 20, 10 + r * 20, os.str());
                        }
                        else {
                                set_color(0x0000f0);
                                draw_box(2 + c * 20, 2 + r * 20, 16, 16);
                        }
                }
        }
        void draw()
        {
                for (size_t i = 0; i < szz; i ++) draw(i);
        }
        bool relabelled;
public:
        bool step()
        {
                do {
                        if (scan >= szz) {
                                if (relabelled) return false;
                                relabel(); relabelled = true;
                                draw();
                                return true;
                        }
                        if (sites[scan]) break;
                        scan ++;
                } while (true);
                if (scan % sz1 && sites[scan - 1]) {
                        int l = find_lab(scan - 1);
                        nlab[l] ++; lab[scan] = l;
                        if (scan >= sz1 && sites[scan - sz1]) {
                                int ll = find_lab(scan - sz1);
                                if (ll != l) { // merging
                                        if (ll < l) {
                                                nlab[ll] += nlab[l];
                                                nlab[l] = - ll;
                                        }
                                        else {
                                                nlab[l] += nlab[ll];
                                                nlab[ll] = - l;
                                        }
                                        for (size_t s = 0; s < scan; s ++) {
                                                draw(s);
                                        }
                                }
                        }
                }
                else if (scan >= sz1 && sites[scan - sz1]) {
                        int l = find_lab(scan - sz1);
                        nlab[l] ++; lab[scan] = l;
                }
                else {
                        nlab[lcnt] = 1;
                        lclr[lcnt] = (unsigned long)(rng.uniform() * 0x1000000);
                        lab[scan] = lcnt ++;
                }
                draw(scan); scan ++;
                return true;
        }
```

hk_step.cc 2/2

```
public:
	HKStep(size_t size0, size_t size1, double prob, unsigned long seed) :
		sz0(size0), sz1(size1), szz(sz0 * sz1), sites(new bool [szz])
	{
		rng.init(seed);
		for (size_t i = 0; i < szz; i ++) sites[i] = rng.uniform() < prob;
		// initialize labeling
		lab = new int [szz]; nlab = new int [szz];
		for (size_t i = 0; i < szz; i ++) lab[i] = - 1;
		lcnt = 0; scan = 0; relabelled = false;
		lclr = new unsigned long [szz];
	}
	~HKStep()
	{
		delete [] sites;
		delete [] lab;
		delete [] nlab;
		delete [] lclr;
	}
	size_t relabel()
	{
		int * nn = new int [szz]; int cc = 0;
		for (int i = 0; i < lcnt; i ++) if (nlab[i] > 0) nn[i] = cc ++;
		for (size_t i = 0; i < szz; i ++) {
			int l = find_lab(i); lab[i] = nn[l];
		}
		cc = 0; int bb = 0;
		for (int i = 0; i < lcnt; i ++) if (nlab[i] > 0) {
			if (bb < nlab[i]) bb = nlab[i];
			nlab[cc ++] = nlab[i];
		}
		lcnt = cc;
		return bb;
	}
};

DrawSys * get_sys()
{
	return new HKStep(20, 30, 0.6, 11);
}
```

draw_sys.hh

```
#include <string>
class DrawSys
{
protected:
		void set_color(unsigned long c);
		void draw_box(int x, int y, int w, int h);
		void draw_text(int x, int y, std::string str);
public:
		virtual void draw() = 0;
		virtual bool step() = 0;
};

extern DrawSys * get_sys();
```

自制的繪圖介面

```
#include "draw_sys.hh"
#include <X11/Xlib.h>
#include <X11/Xatom.h>
#include <iostream>
Display * dpy;
Window win;
Atom wm_del_msg;
GC gc;
XFontStruct * fs;
void DrawSys::set_color(unsigned long c)
{
	XSetForeground(dpy, gc, c);
}
void DrawSys::draw_box(int x, int y, int w, int h)
{
	XFillRectangle(dpy, win, gc, x, y, w, h);
}
void DrawSys::draw_text(int x, int y, std::string str)
{
	XTextItem xt;
	xt.chars = (char *) str.c_str();
	xt.nchars = str.size();
	xt.font = None;
	int rdir;
	int rasc;
	int rdes;
	XCharStruct xcs;
	XTextExtents(fs, xt.chars, xt.nchars, & rdir, & rasc, & rdes, & xcs);
	XDrawText(dpy, win, gc, x - xcs.width / 2, y + (rasc + rdes) / 2 - rdes, & xt, 1);
}
```

draw_x11.cc

```
int main()
{
	DrawSys * sys = get_sys();
	dpy = XOpenDisplay(0);
	if (! dpy) return 0;
	win = XCreateSimpleWindow(dpy, DefaultRootWindow(dpy),
			0, 0, 600, 400, 0, 0, 0xffffff);
	gc = XDefaultGC(dpy, 0);
	fs = XQueryFont(dpy, XGContextFromGC(gc));
	wm_del_msg = XInternAtom(dpy, "WM_DELETE_WINDOW", false);
	XSetWMProtocols(dpy, win, &wm_del_msg, 1);
	XMapWindow(dpy, win);
	XFlush(dpy);
	int x11_fd = XConnectionNumber(dpy);
	fd_set in_fds;
	struct timeval tv;
	XSelectInput(dpy, win, ExposureMask |
			ButtonPressMask | StructureNotifyMask);
	do {
		while (XPending(dpy)) {
			XEvent e;
			XNextEvent(dpy, & e);
			if (e.type == (int) ClientMessage &&
			    e.xclient.data.l[0] == (int) wm_del_msg) {
				goto stop_running;
			}
			switch (e.type) {
			case Expose:
				sys->draw();
				break;
			default: /* unknown event type - ignore it. */
				break;
			}
		}
		FD_ZERO(& in_fds);
		FD_SET(x11_fd, & in_fds);
		tv.tv_usec = 300000;
		tv.tv_sec = 0;
		if (! select(x11_fd + 1, & in_fds, 0, 0, & tv)) {
			sys->step();
		}
	} while (true);
	stop_running:
	return 0;
}
```

draw_win.cc 1/2

```
#include "draw_sys.hh"
#include <windows.h>
#include <fstream>
#include <iostream>
HWND hwnd;              /* This is the handle for our window */
HDC hdc;
HBRUSH bsh;
bool bsh_ok = false;
void DrawSys::set_color(unsigned long c)
{
   if (bsh_ok) {
      DeleteObject(bsh);
      bsh_ok = true;
   }
   bsh = CreateSolidBrush((c & 0xff) << 16 | (c >> 16) | (c & 0xff00));
}
void DrawSys::draw_box(int x, int y, int w, int h)
{
   if (! bsh_ok) return;
   RECT r;
   r.left = x;
   r.right = x + w;
   r.top = y;
   r.bottom = y + h;
   FillRect(hdc, & r, bsh);
}
void DrawSys::draw_text(int x, int y, std::string str)
{
   SIZE sz;
   GetTextExtentPoint32(hdc, str.c_str(), str.size(), & sz);
   SetTextAlign(hdc, TA_CENTER);
   SetBkMode(hdc, TRANSPARENT);
   TextOut(hdc, x, y - sz.cy / 2, str.c_str(), str.size());
}
DrawSys * sys = 0;
/*  Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);
/*  Make the class name into a global variable  */
char szClassName[ ] = "WindowsApp";
```

```
/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message,
 WPARAM wParam, LPARAM lParam)
{
   switch (message)              /* handle the messages */
   {
      case WM_PAINT:
         {
            RECT cr;
            GetClientRect(hwnd, & cr);
            std::cerr << cr.left << ' ' << cr.right << ' '
               << cr.top << ' ' << cr.bottom << std::endl;
            PAINTSTRUCT ps;
            hdc = BeginPaint(hwnd, & ps);
            bsh = CreateSolidBrush(RGB(255,255,255));
            bsh_ok = true;
            sys->draw();
            DeleteObject(bsh);
            bsh_ok = false;
            EndPaint(hwnd, & ps);
         }
         break;
      case WM_DESTROY:
         PostQuitMessage (0);  /* send a WM_QUIT to the message queue */
         break;
      default:                    /* for messages that we don't deal with */
         return DefWindowProc (hwnd, message, wParam, lParam);
   }

   return 0;
}
```

draw_win.cc 2/2

```
int WINAPI WinMain (HINSTANCE hThisInstance, HINSTANCE hPrevInstance,
                LPSTR lpszArgument,  int nFunsterStil)
{
   // RedirectIOToConsole();
   WNDCLASSEX wincl;        /* Data structure for the windowclass */

   /* The Window structure */
   wincl.hInstance = hThisInstance;
   wincl.lpszClassName = szClassName;
   wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
   wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
   wincl.cbSize = sizeof (WNDCLASSEX);

   /* Use default icon and mouse-pointer */
   wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
   wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
   wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
   wincl.lpszMenuName = NULL;                 /* No menu */
   wincl.cbClsExtra = 0;                      /* No extra bytes after the window class */
   wincl.cbWndExtra = 0;                      /* structure or the window instance */
   /* Use Windows's default color as the background of the window */
   wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

   /* Register the window class, and if it fails quit the program */
   if (!RegisterClassEx (&wincl))  return 0;

   /* The class is registered, let's create the program*/
   hwnd = CreateWindowEx (
         0,                   /* Extended possibilites for variation */
         szClassName,         /* Classname */
         "Windows App",       /* Title Text */
         WS_OVERLAPPEDWINDOW, /* default window */
         CW_USEDEFAULT,       /* Windows decides the position */
         CW_USEDEFAULT,       /* where the window ends up on the scr
         608,                 /* The programs width */
         427,                 /* and height in pixels */
         HWND_DESKTOP,        /* The window is a child-window to desk
         NULL,                /* No menu */
         hThisInstance,       /* Program Instance handler */
         NULL                 /* No Window Creation data */
         );

   /* Make the window visible on the screen */
   ShowWindow (hwnd, nFunsterStil);
```

```
   sys = get_sys();
   while (true) {
      MSG msg;
      while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
      {
         if (msg.message == WM_QUIT) return msg.wParam;
         TranslateMessage(&msg);
         DispatchMessage(&msg);
      }
      DWORD r = MsgWaitForMultipleObjects(0, 0, false, 200, QS_ALLINPUT);
      if (r == WAIT_TIMEOUT) {
         hdc = GetDC(hwnd);
         bsh = CreateSolidBrush(RGB(255,255,255));
         bsh_ok = true;
         sys->step();
         DeleteObject(bsh);
         bsh_ok = false;
         ReleaseDC(hwnd, hdc);
      }
   }
}
```

期末專題格式要求

程式碼列表：結構、邏輯清楚，適切縮排。

報告本文（不計圖表 2~3 頁）：題目說明，理論分析，計算方法，數據分析，結果說明

其他文字：程式碼解釋，參考資料，使用工具，貢獻認謝

上傳程式碼到 CP1 SSH 伺服器，放在自己帳戶的 home directory 下名為 final_code 的目錄。並附 Makefile，程式須可以 make 指令編譯出執行檔。

所有報告文字、圖表、程式列表請組合編成一個單一的 PDF 檔。

專題例題

- Percolation on triangle lattice
- Scaling of self-avoiding walk in 3D
- Self-avoiding walk with interaction
- Ising model scaling exponents
- 2D Schrödinger eq. with a potential well
- Simulation of solar system

期末專題期限

- 選定題目，交計劃大綱： 5 月 30 日
- 程式技術咨詢： 5 月 30 日～6 月 10 日
- 完成程式，交初稿： 6 月 13 日
- 完稿繳交期限： 6 月 20 日