Guion de prácticas

Proyecto Final
Instrucciones para la Entrega

DECSAI

Metodología de la Programación

Curso 2018/2019

Instrucciones para la entrega

Descargue de PRADO el fichero `main.cpp`. Dicho fichero debe reemplazar al programa principal de su proyecto. Realice las adaptaciones necesarias (por ejemplo en los nombres de los métodos) para que se puedan probar sus clases.

A través de PRADO, se entregará un unico fichero `proyecto.zip` que contenga el proyecto completo de NetBeans (antes de comprimir, ejecute `make clean` desde la consola).

Todo el código debe estar correctamente modularizado y organizado en las carpetas `src, include`. Cada clase debe tener un fichero `.h` y el correspondiente `.cpp`. En la Fig. 1 puede ver el aspecto que debería tener su estructura de ficheros.

Además, debe crear una carpeta `doc` con un documento `notas.pdf` que contenga:

- Nombre, apellidos y grupo de prácticas (martes, miércoles o viernes). Si ha realizado la práctica en pareja, AMBOS deben entregar la práctica y los dos nombres deben aparecer en el fichero `notas.pdf`.
- Una descripción sobre el desarrollo de la práctica (problemas encontrados, dificultad, sugerencias, qué le ha faltado, etc.).
- Una explicación detallada de las tareas adicionales elegidas (agregado/borrado de partículas por ejemplo).
- Si ha agregado una nueva opción a `main.cpp`, debe explicarla en este documento.
- Si conoce la herramienta `Doxygen` para generar la documentación de las clases, puede utilizarla.
- Para cada una de los casos de prueba, incluya una captura de pantalla del resultado de valgrind al ejecutarlo de la siguiente manera:

```
valgrind --leak-check=full <programa>
```

Si ha realizado alguna tarea que no se haya considerado en los casos anteriores, agregue el código necesario como una nueva opción.

Si para compilar sus programas hace falta alguna instrucción particular, indíquelo también en este fichero.

Al final del documento tiene que indicar una nota (*suspenso, aprobado, notable, sobresaliente*) de auto-evaluación del proyecto final.

El proyecto se compilará utilizando el `Makefile` generado por NetBeans. El programa generado, en función de una opción que se lee por teclado, ejecutará uno de los casos de prueba mostrados en la Fig. 2.

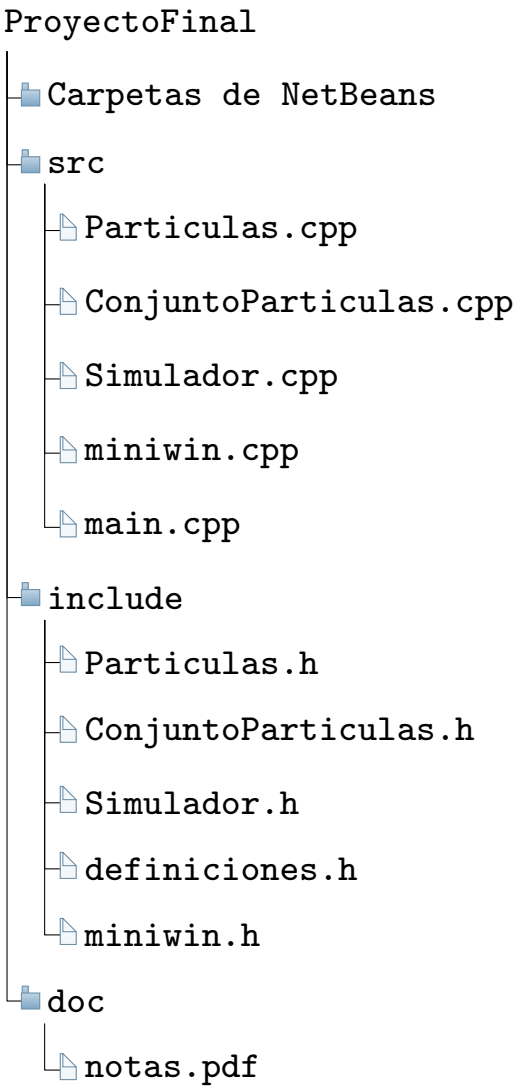

Figura 1: Esquema de carpetas y ficheros a entregar.

Informacion adicional

Sobre Miniwin: La biblioteca `MiniWin` produce una posible pequeña perdida de memoria. Es probable que en sus pruebas obtenga una salida como la mostrada en la Figura 3 (con errores). Compruebe si esos errores son independientes del número de partículas que esté gestionando. Si es así, entonces los errores son de `Miniwin` y no de su programa.

Sobre las instrucciones: Las instrucciones de este documento deben complementarse con las indicaciones y sugerencias dadas en las clases de práctica.

Fecha de Entrega: La fecha límite para la entrega es: Domingo 2 de Junio hasta las 23:59 hs.

```
vredimensiona(100, 100);
ConjuntoParticulas a1, b1(4);
cout << b1 << endl;

Particula x;
a1.AgregaParticula(x);
x.Mover(ancho,alto);
a1.AgregaParticula(x);
cout << a1 << endl;

b1.Mover(ancho, alto);
a1 = a1 + b1;
cout << a1 << endl;

ConjuntoParticulas cc = a1 + b1;
ConjuntoParticulas xx(cc);
cout << xx << endl;
cout << endl;
vcierra();
```

```
ConjuntoParticulas base(50);
ConjuntoParticulas otro(17);
ConjuntoParticulas aux;
int contador = 0;

Simulador game(base, otro, ancho, alto);

while (tecla() != ESCAPE) {
    game.Step();
    game.Pintar();
}
```

```
int NRO = 20;
vredimensiona(ancho, alto);
ConjuntoParticulas rojas(NRO);
ConjuntoParticulas verdes(NRO);
ConjuntoParticulas blancas;
Particula p;

for(int i = 0; i < NRO; i++){
    p = rojas.ObtieneParticula(i);
    p.setColor(ROJO);
    rojas.ReemplazaParticula(i, p);

    p = verdes.ObtieneParticula(i);
    p.setColor(VERDE);
    verdes.ReemplazaParticula(i, p);
}

ConjuntoParticulas aux;
int r1, r2;
for(int i = 0; i < 200; i++){
    rojas.Mover(ancho, alto);
    verdes.Mover(ancho, alto);
    blancas.Mover(ancho, alto);
    aux = rojas + verdes + blancas;

    borra();
    pintaNube(aux);

    if (i % 10 == 0){
        Particula p;
        p.setColor(BLANCO);
        blancas.AgregaParticula(p);
        r1 = rand()%rojas.GetNroParticulas();
        r2 = rand()%verdes.GetNroParticulas()
        rojas.BorraParticula(r1);
        verdes.BorraParticula(r2);
       }
 refresca();
 espera(45);
}
vcierra();
```

Figura 2: Casos de prueba para evaluar las clases implementadas.

```
==3349== HEAP SUMMARY:
==3349==     in use at exit: 65,895 bytes in 70 blocks
==3349==   total heap usage: 614 allocs, 544 frees, 149,884 bytes allocated
==3349==
==3349== 288 bytes in 1 blocks are possibly lost in loss record 27 of 39
==3349==    at 0x4C2CC70: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==3349==    by 0x4012FE4: allocate_dtv (dl-tls.c:296)
==3349==    by 0x4012FE4: _dl_allocate_tls (dl-tls.c:460)
==3349==    by 0x5994D92: allocate_stack (allocatestack.c:589)
==3349==    by 0x5994D92: pthread_create@@GLIBC_2.2.5 (pthread_create.c:500)
==3349==    by 0x403A96: _maybe_call_main() (miniwin.cpp:673)
==3349==    by 0x403AEE: _process_event() (miniwin.cpp:689)
==3349==    by 0x403BD4: main (miniwin.cpp:733)
==3349==
==3349== LEAK SUMMARY:
==3349==    definitely lost: 0 bytes in 0 blocks
==3349==    indirectly lost: 0 bytes in 0 blocks
==3349==      possibly lost: 288 bytes in 1 blocks
==3349==    still reachable: 65,607 bytes in 69 blocks
==3349==         suppressed: 0 bytes in 0 blocks
==3349== Reachable blocks (those to which a pointer was found) are not shown.
==3349== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==3349==
==3349== For counts of detected and suppressed errors, rerun with: -v
==3349== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 114 from 1)
```

Figura 3: Ejemplo de salida de Valgrind. Los errores (en rojo) que aparecen se deben a la biblioteca MiniWin. En verde, se indica que no hay memoria perdida.