Primera Practica: Semáforos

Análisis y Diseño

Alumno: Javier Gutierrez Rodriguez

Usuario ILIAS: jgr00053

Proyecto Gitlab: http://suleiman.ujaen.es:8011/jgr00053/primera-practica

Análisis

Datos

Tipos de datos necesarios para la solución de la practica

- TDA Lista_Escenas
    - Almacena en orden FIFO
    - Operaciones:
        - add() : Inserta al final una escena pasada por parametro
        - remove() : devuelve y elimina el primer elemento de lista
        - find() : devuelve TRUE si la escena se encuentra en la lista y FALSE si no
        - size() : devuelve el tamaño de la lista
- TDA Fotograma:
    - Atributos:
        - idFotograma : String
        - MIN_TIEMPO_FOTOGRAMA : constante entero
        - VARIACION_TIEMPO_FOTOGRAMA : constante entero
        - tiempo_fotograma : entero
    - Operaciones:
        - calculoTiempo() : Calcula un tiempo aleatorio del fotograma entre las constantes
- TDA Escena:
    - Atributos:
        - idEscena : String
        - prioridad (BAJA, ALTA) : Enumerado
        - MIN_NUM_FOTOGRAMA : constante entero
        - VARIACION_NUM_FOTOGRAMA : constante entero
        - num_fotogramas : entero
        - lista_fotogramas : ArrayList(Fotogramas)
        - TIEMPO_FINALIZACION_ESCENA : constantes entero
        - tiempo_renderizado : entero
        - hora_generada : Date
        - hora_comienzo_renderizado : Date
        - hora_fin_renderizado : Date
        - tiempo_procesamiento : Date
    - Operaciones:
        - calculoNumFotogramas() : Calcula el numero de fotogramas de la escena
        - caluloTiempoRenderizado() : Calcula el tiempo en renderizar una escena
        - tiempoProcesamiento() : Calcula el tiempo empleado en el procesamiento
        - setHoraComienzoRenderizado() : inicizializa hora_comienzo_renderizado con la hora pasada
        - setHoraFinRenderizado() : inicizializa hora_fin_renderizado con la hora pasada
        - mostrarInfo() : muestra la informacion de la escena

- TDA Generador de escenas:
    - Atributos:
        - MIN_TIEMPO_GENERACION : constante entero
        - VARIACION_TIEMPO_GENERACION : constante entero
        - numero_escenas : entero
        - tiempo_generacion : entero
    - Operaciones:
        - calculoTiempo() : Calcula el tiempo en generar la escena basandose en las constantes
        - generarEscena() : Genera una escena y la devuelve.
- TDA Renderizador de Escenas:
    - Atributos:
        - lista_resultados : lista de escenas ya renderizadas
    - Operaciones:
        - renderizar() : renderizar las escenas
        - getListaResultados : devuelve la lista de resultados

Variables compartidas

Las variables compartidas entre los Generadores de escenas y los renderizadores son las siguientes:

- escenas_prioritarias : Lista_Escenas que almacena escenas con prioridad alta
- escenas_no_prioritarias : Lista_Escenas que almacena escenas con prioridad baja
- MAX_ESCENAS_EN_ESPERA : Tamaño máximo de las listas.

Semáforos

Los semaforos para solucionar el ejercicio son:

- vacios_escenas_prioritarias y vacios_escenas_no_prioritarias : espacios vacios en las listas. Se inicializan a MAX_ESCENAS_EN_ESPERA.
- llenos_escenas_prioritarias y llenos_escenas_no_prioritarias : espacios llenos en las listas. Se inicializan a cero
- mutex_escenas_prioritarias y mutex_escenas_no_prioritarias : garantizan el acceso seguro a Lista_Escenas. Se inicializan a 1

Procedimientos de apoyo

Para el proceso principal necesitamos:

- crearGeneradores() : Construye NUM_GENERADORES generadores de escenas y se devolvera como arraylist de hilos de generadores
- crearRenderizadores() : Construye NUM_RENDERIZADORES renderizadores de escenas y se devolvera como arraylist de hilos de renderizadores

- ejecutarProcesos() : ejecuta un arraylist pasado como parametro
- comprobacionGeneradores() : comprueba si los generadores han generado todas sus escenas, devuelve un bool
- comprobacionRenderizadores() : comprueba si se ha cumplido comprobacionGeneradores y si no queda ninguna escena en las dos listas
- finalizarProcesos() : finaliza un arraylist de hilos
- mostrarInfo() : muestra la informacion de un array de generadores y otro de renderizadores pasados como argumento. En concreto muestra: numero de escenas procesadas, tiempo total en renderizar todas las escenas, y para cada escena: la hora en la que se genero, la hora en el que un renderizador comenzo su procesamiento, fotogramas que lo componen y su duracion total, la hora en la que el renderizador termino su procesamiento y el tiempo total empleado en el procesamiento de la escena

Diseño

Hilo Principal

```
generadores = crearGeneradores();
renderizadores = crearRenderizadores();
ejecutarProcesos(generadores);
ejecutarProcesos(renderizadores);

if(comprobacionGeneradores())
  finalizarProcesos(generadores);

if(comprobacionRenderizadores())
  finalizarProcesos(renderizadores);

mostrarInfo();
```

Generador de escenas (numero_escenas)

```
Variables locales:
   MIN_TIEMPO_GENERACION : constante entero
   VARIACION_TIEMPO_GENERACION : constante entero
   numero_escenas : entero
   tiempo_generacion : entero
```

```
ejecucionProceso(){
    inicializacionDatos(numero_escenas);
    ejecucion();
  }

  inicializacionDatos(numero_escenas){
    tiempo_generacion = aleatorio(VARIACION_TIEMPO_GENERACION) +
MIN_TIEMPO_GENERACION;
    numero_escenas = numero_escenas;
  }

  ejecucion(){}
    num : entero inicializado a 0
    while(num < numero_escenas){
      escena = generarEscena();

      if(escena.proridad == ALTA){
        wait(vacios_escenas_prioritarias);
        wait(mutex_escenas_prioritarias);

        duerme(tiempo_generacion);
        escenas_prioritarias.add(escena);

        signal(mutex_escenas_prioritarias)
        signal(llenos_escenas_prioritarias)
      }

      else{
        wait(vacios_escenas_no_prioritarias);
        wait(mutex_escenas_no_prioritarias);

        duerme(tiempo_generacion);
        escenas_no_prioritarias.add(escena);

        signal(mutex_escenas_no_prioritarias)
        signal(llenos_escenas_no_prioritarias)
      }
    }
  }
```

Renderizador de escenas

```
  Variables locales:
    lista_resultados : lista de escenas

  ejecucionProceso(){
    inicializacionDatos();
    ejecucion();
  }

  inicializacionDatos(){
    lista_resultados = nueva lista de escenas;
  }
```

```
ejecucion(){
  escena: escena

  while(comprobacionRenderizadores()){
    if(escenas_prioritarias.size > 0){
      wait(llenos_escenas_prioritarias);
      wait(mutex_escenas_prioritarias);

      escena = escenas_prioritarias.remove();
      renderizar(escena);
      lista_resultados.add(escena);

      signal(mutex_escenas_prioritarias);
      signal(vacios_escenas_prioritarias);
    }
    else{
      wait(llenos_escenas_no_prioritarias);
      wait(mutex_escenas_no_prioritarias);

      escena = escenas_no_prioritarias.remove();
      renderizar(escena);
      lista_resultados.add(escena);

      signal(mutex_escenas_no_prioritarias);
      signal(vacios_escenas_no_prioritarias);
    }
  }
}


renderizar(escena){
  escena.setHoraComienzoRenderizado(new Date)
  duerme(escena.tiempo_finalizacion);
  escena.setHoraFinRenderizado(new Date);
}
```