Spesifikasi Tugas Besar
IF2210 - Pemrograman Berorientasi Objek
Semester 2 2017/2018
ArkavQuarium

Pendahuluan

Insaniquarium adalah sebuah *game* di mana kita dapat memelihara ikan yang akan menghasilkan uang yang dapat dikumpulkan. Ada berbagai jenis ikan yang dapat dipelihara di antaranya adalah *guppy*, piranha, dan sebagainya. Ikan-ikan tersebut harus diberi makan dan dirawat agar tidak mati. Selain ikan yang dapat dipelihara ada juga *pet* yang dapat membantu dalam memelihara ikan atau mengumpulkan uang seperti siput, kerang, dll. Kita juga harus melindungi ikan di dalam akuarium dari serangan alien yang dapat muncul sewaktu-waktu.

Gambar 1. Permainan *Insaniquarium*

Pada tugas kecil kali ini anda diminta untuk mengimplementasikan simulasi permainan Insaniquarium bernama **ArkavQuarium** sesuai desain yang kalian . Simulasi ini harus **menghasilkan gambar**, dan disediakan library yang dapat melakukan hal tersebut dengan mudah.

Deskripsi ArkavQuarium

Pada program ini, terdapat sebuah akuarium yang berisi ikan dan entitas-entitas lainnya.

1. Akuarium
    - Akuarium dapat dianggap sebagai sebuah matriks 2D (sumbu X dan Y) dengan ukuran yang sama dengan ukuran layar.
    - Posisi dinyatakan dalam notasi (x, y), di mana x merupakan posisi pada sumbu X/horizontal dan y merupakan posisi pada sumbu Y/vertikal.
    - Posisi ujung kiri-atas akuarium memiliki koordinat (0,0). Semakin ke bawah, maka nilai y semakin besar. Semakin ke kanan, nilai x semakin membesar.
    - Setiap entitas ikan/makanan ikan/koin/piranha yang berada di akuarium harus dicatat posisinya.
2. Guppy
    - Guppy dapat dibeli oleh pemain dengan uang (tentukan harganya sendiri).
    - Ketika guppy dibeli, guppy dimunculkan pada posisi *random* dan dalam kondisi tahap pertumbuhan ke-1.
    - Guppy dimunculkan pada posisi *random* dan dalam kondisi tahap pertumbuhan ke-1.
    - Guppy memiliki **tiga tahap pertumbuhan (1-3).**
    - Guppy dapat pindah dari tahap pertumbuhan lebih kecil ke tahap pertumbuhan lebih besar **setelah memakan sejumlah makanan**.
    - Guppy hanya dapat memakan makanan yang ada di dekatnya (makanan ada di radius tertentu dari guppy).
    - **Setiap C waktu, guppy akan mengeluarkan koin**. Nilai koin harus berbeda pada setiap tahapan guppy, dan guppy yang lebih besar harus mengeluarkan koin dengan nilai lebih besar.
    - Jika tahap pertumbuhan guppy sudah tahap ke-3, maka ikan tidak dapat tumbuh ke tahap lebih tinggi.
    - Guppy dapat bergerak ke arah apapun (360 derajat). Gerakan guppy memiliki kecepatan tetap tanpa memperhatikan arah gerak guppy (guppy tidak boleh bergerak lebih cepat ketika bergerak diagonal, atau sebaliknya).
    - Guppy tidak boleh bergerak keluar dari akuarium. Jika guppy sedang berjalan-jalan dengan arah acak, pilih arah baru yang tidak akan mengeluarkan guppy dari akuarium.
    - Tampilan guppy harus mengikuti arah gerak pada sumbu horizontal. Jika guppy sedang bergerak ke kanan (atau kanan atas, kanan bawah, dst..), maka tampilan guppy adalah melihat ke kanan, dan sebaliknya. Gerakan pada sumbu vertikal tidak mempengaruhi tampilan.
    - Tampilan guppy harus berbeda untuk setiap tahap pertumbuhan.

- Guppy memiliki dua kondisi, lapar dan kenyang. Guppy dibuat dalam kondisi kenyang.
- **Dalam keadaan kenyang**, guppy **bergerak dengan arah acak selama T waktu**. Pada periode ini, guppy tidak mungkin memakan makanan.
- Setelah T waktu, guppy berpindah ke keadaan lapar dan membutuhkan makanan yang harus dipenuhi **dalam waktu M**. Jika ada makanan guppy di dalam akuarium, dia akan bergerak ke arah **makanan terdekat**. Jika tidak ada makanan, maka guppy bergerak dengan arah acak seperti keadaan kenyang.
- Jika makanan guppy yang sedang didekati hilang (dimakan guppy lain atau jatuh ke bawah), guppy mencari makanan lain jika ada, atau bergerak dengan arah acak jika tidak ada.
- Perilaku **guppy bergerak dengan arah acak** adalah sebagai berikut:
  - Guppy memilih suatu arah secara acak dan bergerak dengan arah tersebut selama waktu X.
  - Setelah waktu X, guppy memilih arah baru, dan bergerak dengan arah tersebut selama waktu X.
  - Gerakan tersebut diulang hingga guppy memiliki arah gerak lain.
  - Waktu X bebas, namun harus di atas 0.5 detik dan di bawah 5 detik.
- Jika dalam waktu M guppy tidak menemukan makanan, guppy mati.

3. Makanan Ikan
   - Makanan ikan muncul **ketika dibeli oleh pemain** (harus mengurangi uang), dan muncul di tempat yang **dapat dipilih** oleh pemain (contoh: terdapat *cursor* yang dapat digerakkan pemain dengan keyboard).
   - Makanan ikan selalu muncul di atas akuarium.
   - Makanan ikan selalu bergerak ke bawah.
   - Kecepatan gerak makanan ikan harus lebih lambat dari kecepatan ikan.
   - Ketika makanan ikan sampai di dasar, **makanan tersebut hilang**.
4. Koin
   - Koin memiliki suatu nilai.
   - Koin selalu bergerak ke bawah.
   - Ketika koin sampai di dasar, koin tersebut diam.
   - Koin diambil oleh siput dan salah satu spek bonus.
   - Koin yang diambil meningkatkan nilai atribut uang pemain sejumlah nilai koin.
5. Piranha
   - Piranha pada dasarnya sama seperti ikan (periode makan, cara bergerak, mati kelaparan, dst.). Perbedaannya akan dijelaskan pada poin-poin selanjutnya.
   - Piranha **tidak memakan makanan ikan**. Makanan piranha adalah ikan dengan tahap apapun.
   - Koin yang dihasilkan piranha tidak dihasilkan secara periodik, tetapi **langsung muncul** setelah dia makan. Nilai koin adalah **harga ikan * (tahap ikan yang dimakan + 1)**.
6. Siput
   - Permainan mulai dengan satu siput.

- Siput hanya ada di dasar akuarium.
- Siput bergerak ke arah koin terdekat yang ada di dasar.
- Tampilan siput harus mengikuti arah gerak.
- Jika tidak ada koin yang di dasar, siput bergerak ke arah koin yang paling dekat ke dasar.
- Jika tidak ada koin, siput diam.
- Siput mengambil koin yang ada di dekatnya (koin ada di radius tertentu dari siput).

Hal-hal lain yang harus ada, namun tidak berupa entitas di dalam akuarium, adalah:

1. Uang
    - Uang merupakan nilai *integer* yang menyatakan jumlah uang yang dimiliki oleh pemain pada suatu waktu.
    - Jumlah uang harus selalu ditampilkan.
    - Pada awal permainan, pemain memiliki sejumlah uang (tidak 0).
2. Telur
    - Seperti *item* lainnya, telur dapat dibeli dengan menggunakan sejumlah uang.
    - Ketika membeli telur, telur tidak perlu ditampilkan pada dunia akuarium. Anda cukup memperlihatkan jumlah telur yang sudah pernah dibeli pengguna.
3. Kondisi Menang/Kalah
    - Jika jumlah telur yang dibeli pengguna sudah mencapai 3, permainan berakhir dengan status menang.
    - Jika pengguna tidak memiliki ikan apapun dan uang pemain tidak memungkinkan lagi untuk membeli ikan, permainan berakhir dengan status kalah.
4. Informasi permainan
    - Tombol-tombol yang digunakan untuk melakukan semua aksi harus ditampilkan.

Spesifikasi

Berikut adalah spesifikasi dari tugas kecil ini.

1. Tugas ini **dikerjakan berkelompok** yang sama dengan tugas kecil.
2. Buatlah implementasi seluruh deskripsi ArkavQuarium yang telah dijelaskan sebelumnya dengan **pendekatan berorientasi objek** (classes dan objects).
3. Implementasi yang dibuat disesuaikan dengan desain yang telah dibuat sebelumnya. Semua perubahan pada desain (termasuk kelas tambahan) harus dituliskan dan disertai alasan.
4. Program menampilkan kondisi akuarium **dalam bentuk gambar**. Pilihlah gambar-gambar untuk setiap entitas yang cocok. **Disarankan menggunakan library yang disediakan**, namun tidak diwajibkan. Anda boleh mengubah kode library yang disediakan.
5. Jika ingin menggunakan library lain, harus ditanyakan terlebih dahulu melalui metode tanya jawab yang disediakan.

6. Anda boleh menambahkan hal-hal yang tidak ada di spek, selama tidak bertentangan dengan spek wajib.
    a. Contoh tambahan yang diperbolehkan: siput menjadi bisa dibeli
    b. Contoh tambahan yang **tidak** diperbolehkan: siput bisa bergerak ke atas dari dasar akuarium
7. Untuk setiap kelas, implementasikan ***driver*** untuk **menguji** kelas tersebut.
8. Buatlah **laporan tugas besar** dengan **isi sesuai template** yang disediakan.
9. Selain kode program dan laporan, cantumkan **dokumentasi program** yang telah **di-*generate* dengan doxygen**.
10. Anda harus **melakukan asistensi**, yang dijelaskan di bawah.
11. Program yang Anda buat tetap mempunyai elemen pemrograman berorientasi objek di bawah ini.
    a. Inheritance (“Is-a” relationship)
    b. Method override pada inheritance
    c. Aggregation (“Has-a” relationship)
    d. Polymorphism
    e. Generic
    f. Abstract class dan interface
    g. Function/method overloading
    h. Interface (diimplementasikan sebagai abstract class tanpa implementasi)
    i. Multiple inheritance (disarankan satu class dan satu atau lebih interface)
12. Manfaatkan elemen-elemen pemrograman berorientasi objek tersebut untuk membuat program yang baik, seperti:
    a. Tidak terdapat kode duplikat
    b. Memiliki struktur kelas yang mudah dipahami
    c. Implementasi kelas tidak terlalu kompleks
    d. Mengikuti prinsip SOLID.
13. Selain entitas-entitas yang sudah disebutkan sebelumnya, program yang Anda buat **diwajibkan mengimplementasikan kelas *linked list*** yang dapat menyimpan **tipe *generic*.**
14. Jika linked list ini digunakan di minimal dua tempat dengan tipe yang berbeda, maka linked list memenuhi spek desain generic.
15. Berikut adalah method *linked list* **minimal** (boleh Anda tambah sesuai kebutuhan) yang harus dibuat.
    a. find(T element) → int
       Method ini mengembalikan indeks di mana elemen berada pada *linked list*, dan -1 jika tidak ada.
    b. isEmpty()
       Method ini mengembalikan nilai True jika *linked list* kosong, dan False jika sebaliknya.
    c. add(T element) → void
       Method ini menambahkan elemen sebagai elemen paling belakang pada LinkedList

d. remove(T element) → void
   Method ini membuang elemen dengan identitas demikian
e. get(int index) → T
   Method ini mengembalikan elemen dengan tipe T pada indeks ke-i.

Bonus

- Hiaslah program yang kalian buat sesuai kreativitas kalian (tambah background, entitas hiasan, dan lain-lain).
- Buat suatu *main menu* dari program.
- Fungsi save dan load.
- Penentuan posisi makanan ikan dilakukan menggunakan *mouse*. Lakukan eksplorasi terhadap library yang disediakan.
- Pemain dapat mengambil koin menggunakan *mouse*.

Deliverable dan Deadline

Deliverable yang harus dikirimkan adalah:

1. Folder *src* yang berisi:
   a. *Source code* yang telah dibuat
   b. *Makefile* untuk kompilasi program
2. Folder doc yang berisi:
   a. Laporan **dalam format .pdf** sesuai template laporan. Dinamakan **Laporan.pdf**.
   b. Folder dokumentasi yang dibangkitkan oleh *tools* (e.g. **doxygen**). Biasanya dalam format html. Beri nama folder **documentation**.
   c. Readme.txt yang berisi cara menjalankan program dan hal-hal yang harus diperhatikan

Semua deliverable tersebut di-zip dengan format nama:

TubesOOP_<NIM1>_<NIM2>_<NIM3>_<NIM4>.zip

Deliverable dikumpulkan selambat-lambatnya pada hari **Rabu, 11 April 2018 pukul 23.59** melalui *uploader* di Olympia.

Perhatikan bahwa **Anda akan mendemokan apa yang Anda submit terakhir di *deadline***. Sebagai gambaran, saat demo, peserta akan diberikan berkas hasil *submission* oleh asisten dan harus di-*build* ulang di komputer peserta. Jadi, pastikan minimal yang Anda submit adalah **program yang jalan**. Jika yang Anda submit terakhir tidak dapat dijalankan, Anda dipastikan mendapat nilai 0.

Asistensi

1. Anda **wajib** melakukan asistensi **minimal 1x** di luar kelas dengan asisten. Daftar asisten dapat dilihat di Olympia.

2. Salah satu asistensi harus dilakukan **sebelum tanggal 7 April 2018**. Namun Anda boleh melakukan asistensi lagi setelah tanggal tersebut.
3. Jika ingin melakukan asistensi, hubungi asisten minimal 1 hari sebelum melakukan asistensi.

Tips-Tips

- Gerakan ikan dapat diimplementasikan sebagai perhitungan berikut setiap frame/iterasi main loop:
  x' = x + v*cos(a)*t
  y' = y + v*sin(a)*t
  Di mana x', y' adalah posisi sesudah, x, y adalah posisi sebelum, v adalah kecepatan, a adalah arah gerak dalam radian, dan t adalah perbedaan waktu dari iterasi sebelumnya.
- Arah gerak ikan ke suatu titik dapat dihitung dengan menggunakan fungsi atan2 sebagai berikut:
  a = atan2(y2 - y1, x2 - x1)
  Di mana x1, y1 dan x2, y2 adalah posisi dua objek dan a adalah suatu arah.
  Jika objek yang berada di x1, y1 bergerak dengan arah a, maka dia akan mendekati x2, y2.
- Untuk melakukan aksi yang berlangsung selama beberapa waktu, dapat disimpan waktu aksi tersebut mulai, dan mengecek apakah perbedaan waktu mulai dengan waktu sekarang sudah lebih besar dari durasi yang diinginkan pada iterasi selanjutnya.