The Purge

Albert Oliveras

18 de desembre de 2020

1 Regles del Joc

Per tal de reduir el nombre de crims al país, el règim polític anomenat *La Nova Fundació dels Pares d'Amèrica* ha implementat una mesura catàrtica davant l'imperant violència que regna als carrers: la *Purga Anual*. Cada any, durant una sèrie de dies, es pot cometre qualsevol crim durant la nit sense haver de respondre davant la justícia. El país s'ha organitzat en clans de ciutadans afins que intenten sobreviure.

Es tracta d'un joc per a quatre jugadors, identificats amb números de 0 a 3. Cada jugador té el control d'un clan format per guerrers i constructors. Cadascun d'ells, que anomenarem de forma general "ciutadà", neix amb uns determinats punts de vida.

El joc dura un cert nombre de jornades, i cada jornada comença amb un nombre N de rondes de dia, durant les quals no es permeten els crims, i continua amb N rondes de nit durant les quals els crims són impunes. Hi ha elements del joc que tenen un comportament diferent de dia i de nit. A cada ronda, cada membre dels clans pot efectuar com a molt una acció. Com a resultat d'aquestes accions, els clans poden guanyar punts. Més concretament, es guanyen punts en recollir diners o matar a ciutadans rivals. El guanyador de la partida és el clan que obtingui més punts.

El tauler del joc és un rectangle $N \times M$ del qual ningú en pot sortir, que representa una ciutat. Una posició del tauler ve determinada per un parell d'enters (f, c) on $0 \leq f < N$ i $0 \leq c < M$. La posició de més a dalt i a l'esquerra és la $(0, 0)$, mentre que la de més a baix i a la dreta és $(N-1, M-1)$. Cada cel·la del tauler o bé forma part d'un carrer o bé d'un edifici. Els ciutadans no poden trepitjar cap edifici i s'han de moure pels carrers. En els carrers, s'hi poden trobar armes, diners i menjar. De cara a protegir-se dels possible atacs, s'hi poden construir barricades.

Barricades. Els únics que poden construir barricades són els constructors, però només durant el dia. Des de la seva posició, poden crear una barricada a una de les com a molt 4 cel·les adjacents en vertical o horitzontal, sempre i quan estiguin buides. Si un constructor està amagat en un barricada, no podrà construir-ne una altra a cap cel·la adjacent. Les barricades podran ser ocupades per qualsevol ciutadà del clan que l'ha creat, però no pels clans contraris. Les barricades tenen una resistència inicial, que pot ser incrementada fins a un cert valor màxim a base de repetides operacions de construcció. Cada clan pot tenir un nombre màxim de barricades. Quan acaba la nit i comença un nou dia, totes les barricades que han resistit desapareixen.

Guerrers. Els guerrers porten sempre una arma. Inicialment tots porten un senzill martell, però poden agafar una pistola o un bazuca. Durant el dia, un guerrer pot moure's pel tauler de la manera següent:

- Només pot accedir a les cel·les adjacents en horitzontal i vertical, mai en diagonal.
- Si es mou a una cel·la ocupada per un edifici, un altre ciutadà o una barricada rival, el moviment s'ignorarà.
- Si es mou a una cel·la ocupada per una barricada del seu clan, ocuparà la cel·la sense destruir la barricada.
- Si es mou a una cel·la ocupada per diners, el seu clan rebrà els punts corresponents i el guerrer ocuparà la cel·la tot fent desaparèixer els diners.
- Si es mou a una cel·la ocupada per menjar, la seva vida s'incrementarà i el guerrer ocuparà la cel·la tot fent desaparèixer el menjar. La vida d'un ciutadà mai podrà superar la seva vida inicial.
- Si es mou a una cel·la ocupada per una arma, el guerrer ocuparà la cel·la i l'arma desapareixerà. Tot seguit, el guerrer passarà a portar l'arma més potent d'entre la que tenia i la que ha trobat. Un bazuca és més potent que una pistola, i aquesta més potent que un martell.

Durant la nit, el comportament d'un guerrer és el mateix excepte quan es mou a una cel·la ocupada per un ciutadà o una barricada rival:

- Si es mou a una cel·la ocupada per un ciutadà del seu clan, res canvia i el moviment s'ignorarà.
- Si es mou a una cel·la ocupada per un ciutadà d'un altre clan, s'iniciarà una lluita. Qui perdi la lluita perdrà un nombre prefixat de punts de vida. El guanyador de la lluita vindrà determinat per la força del dos contrincants. Els constructors sempre tenen la mateixa força, mentre que la força dels guerrers depèn de l'arma que porten. En un atac entre individus amb forces N i M, el primer guanyarà la lluita amb probabilitat $N/(N+M)$ i el segon amb probabilitat $M/(N+M)$. En qualsevol cas, després de la lluita cap dels dos contrincants canviarà de cel·la. No obstant, un d'ells pot morir si es queda sense punts de vida. En aquest cas, el supervivent rebrà una certa quantitat de punts.
- Si es mou a una cel·la ocupada per una barricada rival, la barricada perdrà tants punts de resistència com indiqui la força de demolició de l'arma que porta. En cas que la barricada hagi perdut tota la resistència, desapareixerà, però el guerrer no es mourà encara de cel·la.

Constructors. Els constructors no porten mai cap arma. Durant el dia, els seus moviments són idèntics als moviments dels guerrers. L'única diferència és que quan un constructor es mou a una cel·la ocupada per una arma, el constructor ocuparà la cel·la i l'arma desapareixerà (es perdrà).

Durant la nit, el comportament d'un constructor és el mateix que durant el dia excepte quan:

- Es mou a una cel·la ocupada per un ciutadà rival. En aquest cas, s'inicia una lluita com la ja explicada anteriorment.
- Es mou a una barricada rival. En aquest cas, passa el mateix que amb els guerrers, però considerant la força de demolició que tenen els constructors, que és sempre la mateixa, ben poca!

En general, quan un ciutadà aconsegueix demolir una barricada on s'hi amaga un rival, aquest no mor, sinó que es queda al descobert i ja pot rebre atacs. Si un ciutadà està amagat dins una barricada, ningú put iniciar un atac contra ell. En canvi, ell sí pot iniciar un atac contra un altre ciutadà. Durant aquest atac, no tindrà cap protecció especial i per tant, podrà morir.

Regeneració d'objectes. Cada vegada que desapareixen diners o menjar, aquests reapareixen al cap d'un cert nombre de rondes. Les armes i ciutadans també reapareixen. Una arma es veu reemplaçada per una del mateix tipus. Els ciutadans es reemplacen per ciutadans del mateix tipus amb la corresponent vida inicial. Els guerrers reapareixen portant un martell. En tots aquests casos, l'objecte reapareixerà sempre en una cel·la buida C i tal que no hi ha cap ciutadà en les posicions que l'envolten (les marcades amb una x a la taula):

x	x	x	x	x
x	x	x	x	x
x	x	C	x	x
x	x	x	x	x
x	x	x	x	x

Si en el moment de reaparèixer no existeix cap cel·la segura en aquest sentit, l'objecte intentarà reaparèixer a la següent ronda. A cada ronda primer s'intentarà fer reaparèixer tots els ciutadans, a continuació diners i menjar amb la mateixa prioritat, i finalment les armes, totes elles amb la mateixa prioritat. És a dir, si en una ronda ha de reaparèixer un ciutadà i un bonus de diners i només queda una cel·la segura, reapareixerà el ciutadà i els diners esperaran a la següent ronda.

Execució d'ordres. A cada ronda es pot donar més d'una ordre al mateix ciutadà, tot i que només se seleccionarà la primera d'elles (si n'hi ha alguna). Tot programa que intenti donar més de 1000 comandes durant la mateixa ronda s'avortarà.

Cada ronda, les ordres seleccionades dels quatre jugadors s'executaran amb ordre aleatori, però respectant l'ordre relatiu entre els ciutadans d'un mateix clan. Com a conseqüència de la norma anterior, considereu la possibilitat de donar les ordres als vostres ciutadans a cada ronda de més urgent a menys urgent.

Tingueu en compte que s'aplica cada moviment sobre el tauler que resulta dels moviments anteriors. Per exemple, considereu el tauler

x	x	x	x
x	M	C	x
x	G	x	x
x	x	x	x

on M representa menjar, C un constructor i G un guerrer rival. Imaginem que el jugador que controla C ha decidit que vagi cap a l'esquerra, i el jugador que controla G ha decidit que vagi amunt. Si s'executa primer el moviment de G, aleshores C s'ha quedat sense menjar, perquè G ja l'ha consumit i a més, la posterior execució del moviment de C és un atac cap a G del qual segurament en sortirà malparat.

2 El Visor

A la Figura 1 es mostra una captura de pantalla amb tots els elements del joc.

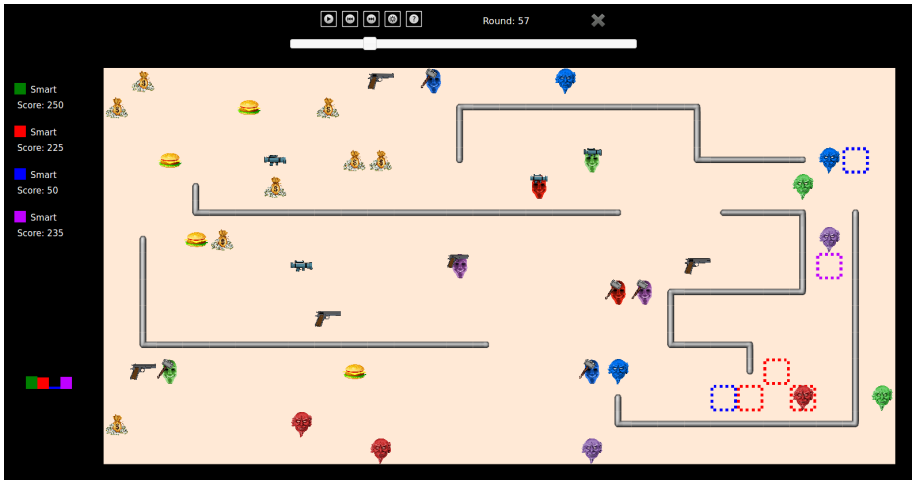

Figura 1: Captura de pantalla del joc.

- A la part superior hi ha botons que permeten reproduir o pausar la partida, anar al començament o al final de la partida, activar o desactivar el mode d'animació o obtenir una finestra d'ajuda amb més maneres de controlar com es reprodueix la partida. També hi trobareu la ronda actual i un botó per tancar el visor. Una barra de desplaçament horitzontal mostra visualment en quin punt de la partida es troba la ronda actual.

- A la columna de l'esquerra, apareix cada jugador amb el nom i color corresponents. A sota es mostren els punts acumulats. A les partides jugades a Jutge.org, també es mostra el percentatge de temps de CPU que s'ha consumit fins ara (si està esgotat, s'indica amb un 'out'). A la part inferior de la columna esquerra hi ha un gràfic de barres que mostra de manera visual els punts de cada jugador.
- Les cel·les dels carrers estan pintades de color marró clar de dia, i de marró fosc de nit. Les cel·les dels edificis estan marcades amb barres de color gris.
- Les armes i els bonus es representen com a la Figura 2.
- Els ciutadans es representen com a la Figura 3.
- Les barricades es representen com a la Figura 4.

Figura 2: Representació d'un bazuca, una pistola, menjar i diners.

Figura 3: Representació d'un constructor i de tres guerrers amb martell, pistola i bazuca, respectivament.

Figura 4: Representació de barricades, de menys resistent (esquerra) a més resistent (dreta).

3 Com programar un jugador

El primer que heu de fer és descarregar-vos el codi font. Aquest inclou un programa C ++ que executa les partides i un visualitzador HTML per veure-les en un format raonable i animat. A més, us proporcionem un jugador "Null" i un jugador "Demo" per facilitar el començament de la codificació del jugador.

3.1 Executar la primera partida

Aquí us explicarem com executar el joc sota Linux, però hauria de funcionar també sota Windows, Mac, FreeBSD, OpenSolaris, ... Només necessiteu una versió recent de `g ++`, el `make` instal·lat al sistema, a més d'un navegador modern com Firefox o Chrome.

1. Obriu una consola i feu `cd` al directori on us heu descarregat el codi font.
2. Si, per exemple, teniu una versió de Linux en 64 bits, executeu:

   ```
   cp AIDummy.o.Linux64 AIDummy.o
   ```

   Amb altres arquitectures, cal escollir els objectes adequats que trobareu al directori.
3. Executeu

   ```
   make all
   ```

   per compilar el joc i tots els jugadors. Tingueu en compte que el `Makefile` identifica com a jugador qualsevol fitxer que coincideixi amb `AI*.cc`
4. Es crea un fitxer executable anomenat `Game`. Aquest executable us permet executar una partida mitjançant una comanda com la següent:

   ```
   ./Game Demo Demo Demo Demo -s 30 < default.cnf > default.res
   ```

   Aquesta comanda comença una partida, amb la llavor aleatòria 30, amb quatre instàncies del jugador Demo, al tauler definit a `default.cnf`. La sortida d'aquesta partida es redirigeix a `default.res`.
5. Per veure una partida, obriu el fitxer visualitzador `Viewer/viewer.html` amb un navegador i carregueu el fitxer `default.res`.

Utilitzeu

```
./Game --help
```

per veure la llista de paràmetres que es poden usar. Particularment útil és

```
./Game --list
```

per veure tots els noms de jugadors reconeguts.

En cas que sigui necessari, recordeu que podeu executar

```
make clean
```

per esborrar l'executable i els objectes i començar la compilació de nou.

3.2 Arxiu de configuració

Us proporcionem dos exemples d'arxius de configuració. En totes les partides que es juguin al Jutge, incloent les eliminatòries i la final, utilitzarem sempre `default.cnf`. Aquest arxiu fixa els paràmetres de la Figura 5 al valor per defecte. La posició dels edificis, dels ciutadans, de les armes i dels bonus es decideixen de forma aleatòria.

L'arxiu de configuració `default-fixed.cnf` mostra com es poden canviar els paràmetres i definir el tauler d'entrada. Els caràcters de la graella són '.'(carrer), 'B' (edifici), 'G' (pistola), 'Z' (bazuca), 'M' (diners), 'F' (menjar). Pel que fa als ciutadans, el seu tipus és 'w' (guerrer) o bé 'b' (constructor), i les armes possibles són 'n' (cap arma), 'h' (martell), 'g' (pistola) i 'b' (bazuca).

Podeu crear arxius de configuració vosaltres mateixos per diversió o si voleu provar els vostres jugadors en taulers més petits o amb alguna peculiaritat. Els valors dels paràmetres han d'estar dins el rang establert a la Figura 5. Heu d'assegurar-vos que les forces d'atac i de demolició del bazuca han de ser majors o iguals que les de la pistola, i les de la pistola majors o iguals que les del martell. Una altra restricció és que BARRICADE_RESISTANCE_STEP ha de ser menor o igual que BARRICADE_MAX_RESISTANCE. Finalment, si voleu modificar els paràmetres i generar el tauler aleatòriament, heu d'assegurar-vos que el tauler és prou gran per encabir tots els objectes.

3.3 Afegir el vostre jugador

Per crear un jugador nou copieu `AINull.cc` (un jugador buit que proporcionem com a plantilla) a un fitxer nou `AIElquesigui.cc`. A continuació, editeu el fitxer nou i canvieu la línia

#define PLAYER_NAME Null

a

#define PLAYER_NAME Elquesigui

Paràmetre	Valor per defecte	Rang
NUM_PLAYERS	4	$[4, 4]$
NUM_DAYS	5	$[1, \infty)$
NUM_ROUNDS_PER_DAY	50	$[2, \infty)$ (parell)
BOARD_ROWS	15	$[12, 25]$
BOARD_COLS	30	$[12, 50]$
NUM_INI_BUILDERS	4	$[1, 6]$
NUM_INI_WARRIORS	2	$[1, 4]$
NUM_INI_MONEY	10	$[0, 10]$
NUM_INI_FOOD	5	$[0, 10]$
NUM_INI_GUNS	4	$[0, 5]$
NUM_INI_BAZOOKAS	2	$[0, 4]$
BUILDER_INI_LIFE	60	$[1, \infty)$
WARRIOR_INI_LIFE	100	$[1, \infty)$
MONEY_POINTS	5	$[1, \infty)$
KILL_BUILDER_POINTS	100	$[1, \infty)$
KILL_WARRIOR_POINTS	250	$[1, \infty)$
FOOD_INCR_LIFE	20	$[1, \infty)$
LIFE_LOST_IN_ATTACK	20	$[1, \infty)$
BUILDER_STRENGTH_ATTACK	1	$[1, \infty)$
HAMMER_STRENGTH_ATTACK	10	$[1, \infty)$
GUN_STRENGTH_ATTACK	100	$[1, \infty)$
BAZOOKA_STRENGTH_ATTACK	1000	$[1, \infty)$
BUILDER_STRENGTH_DEMOLISH	3	$[1, \infty)$
HAMMER_STRENGTH_DEMOLISH	10	$[1, \infty)$
GUN_STRENGTH_DEMOLISH	10	$[1, \infty)$
BAZOOKA_STRENGTH_DEMOLISH	30	$[1, \infty)$
NUM_ROUNDS_REGEN_BUILDER	50	$[1, \infty)$
NUM_ROUNDS_REGEN_WARRIOR	50	$[1, \infty)$
NUM_ROUNDS_REGEN_FOOD	10	$[1, \infty)$
NUM_ROUNDS_REGEN_MONEY	5	$[1, \infty)$
NUM_ROUNDS_REGEN_WEAPON	40	$[1, \infty)$
BARRICADE_RESISTANCE_STEP	40	$[1, \infty)$
BARRICADE_MAX_RESISTANCE	320	$[1, \infty)$
MAX_NUM_BARRICADES	3	$[1, \infty)$

Figura 5: Paràmetres del joc. La seva explicació la trobareu a l'arxiu `Settings.hh`.

El nom que trieu pel vostre jugador ha de ser únic, no ofensiu i tenir com a màxim 12 caràcters. Aquest nom es mostrarà al lloc web i durant les partides.

A continuació, podeu començar a implementar el mètode virtual *play* (), heretat de la classe base *Player*. Aquest mètode, que serà cridat a cada ronda, ha de determinar les ordres que s'enviaran a les vostres unitats.

Podeu utilitzar definicions de tipus, variables i mètodes a la vostra classe de jugador, però el punt d'entrada del vostre codi serà sempre el mètode *play* ().

Des de la vostra classe jugador també podeu cridar funcions que trobareu especificades als arxius següents:

- `State.hh`: accedir a l'estat del joc.
- `Action.hh`: donar ordres als vostres ciutadans.
- `Structs.hh`: estructures de dades útils.
- `Settings.hh`: accedir als paràmetres del joc.
- `Player.hh`: mètode *me*().
- `Random.hh`: generar nombre aleatoris.

També podeu examinar el codi del jugador "Demo" a `AIDemo.cc` com a exemple de com usar aquestes funcions.

Tingueu en compte que no heu d'editar el mètode *factory* () de la classe del vostre jugador, ni l'última línia que afegeix el vostre jugador a la llista de jugadors disponibles.

3.4 Jugar contra el jugador Dummy

Per a provar la vostra estratègia contra el jugador Dummy, proporcionem el seu arxiu objecte. D'aquesta manera, no teniu accés al seu codi font però podreu afegir-lo com a jugador i competir contra ell en local.

Com ja hem comentat, per afegir el jugador Dummy a la llista de jugadors registrats, heu de copiar l'arxiu corresponent a la vostra arquitectura cap a `AIDummy.o`. Per exemple:

```
cp AIDummy.o.Linux64 AIDummy.o
```

Recordeu que els arxius objecte contenen instruccions binàries per a una arquitectura concreta, pel que no podem proporcionar un únic arxiu.

Consell de pro: demaneu als vostres amics els seus arxius **objecte** (mai codi font!!!) i afegiu-los al vostre `Makefile`!

3.5 Restriccions en enviar el vostre jugador

Quan creieu que el vostre jugador és prou fort per entrar a la competició, podeu enviar-lo al Jutge. Degut a que s'executarà en un entorn segur per prevenir trampes, algunes restriccions s'apliquen al vostre codi:

- Tot el vostre codi font ha d'estar en un sol fitxer (com AIElquesigui.cc).
- No podeu utilitzar variables globals (en el seu lloc, utilitzeu atributs a la vostra classe).
- Només teniu permès utilitzar biblioteques estàndard com `iostream`, `vector`, `map`, `set`, `queue`, `algoritme`, `cmath`, ... En molts casos, ni tan sols cal incloure la biblioteca corresponent.
- No podeu obrir fitxers ni fer cap altra crida a sistema (threads, forks, ...)
- El vostre temps de CPU i la memòria que utilitzeu seran limitats, mentre que no ho són al vostre entorn local quan executeu `./Game`. El temps límit és d'un segon per l'execució de tota la partida. Si exhauriu el temps límit (o si l'execució del vostre codi avorta), el vostre jugador es congelarà i no admetrà més instruccions.
- El vostre programa no ha d'escriure a **cout** ni llegir de **cin**. Podeu escriure informació de depuració a **cerr**, però **heu d'eliminar** aquesta informació en el codi que envieu al Jutge.
- Qualsevol enviament al Jutge ha de ser un intent honest de jugar. Qualsevol intent de fer trampes de qualsevol manera serà durament penalitzat.
- Un cop hagueu enviat un jugador al Jutge que hagi derrotat al Dummy, podeu fer més enviaments però haureu de canviar el nom del jugador. És a dir, un cop un jugador ha vençut al Dummy, el seu nom queda bloquejat i no es pot reutilitzar.

4 Consells

- **NO DONEU O DEMANEU EL VOSTRE CODI A NINGÚ**. Ni tan sols una versió antiga. Ni fins i tot al vostre millor amic. Ni tans sols d'estudiants d'anys anteriors. Utilitzem detectors de plagi per comparar els vostres programes, també contra enviaments de jocs d'anys anteriors. No obstant, podeu compartir arxius objecte.

  Qualsevol plagi implicarà **una nota de 0 en l'assignatura** (no només del Joc) de tots els estudiants involucrats. Es podran també prendre mesures disciplinàries addicionals. Si els estudiants A i B es veuen implicats en un plagi, les mesures s'aplicaran als dos, independentment de qui va crear el codi original. No es farà cap excepció sota cap circumstància.

- Abans de competir amb els companys, concentreu-vos en derrotar al Dummy.
- Llegiu les capçaleres de les classes que aneu a utilitzar. No cal que mireu les parts privades o la implementació.
- Comenceu amb estratègies simples, fàcils d'implementar i depurar, ja que és exactament el que necessitareu al principi.
- Definiu mètodes auxiliars senzills (però útils) i *assegureu-vos que funcionin correctament*.
- Intenteu mantenir el vostre codi net. Això farà més fàcil canviar-lo i afegir noves estratègies.
- Com sempre, compileu i proveu el vostre codi sovint. És *molt* més fàcil rastrejar un error quan només heu canviat poques línies de codi.
- Utilitzeu **cerr** per produir informació de depuració i afegiu `asserts` per assegurar-vos que el vostre codi fa el que hauria de fer. No oblideu eliminar-los abans de pujar el codi. En cas de no fer-ho, el vostre jugador morirà.
- Quan depureu un jugador, elimineu els **cerr**s que tingueu en el codi d'altres jugadors, per tal de veure només els missatges que desitgeu.
- Podeu utilitzar comandes com el `grep` de Linux per tal de filtrar la sortida produïda per `Game`.
- Activeu l'opció `DEBUG` al Makefile, que us permetrà obtenir traces útils quan el vostre programa avorta. També hi ha una opció `PROFILE` que podeu utilitzar per optimitzar codi.
- Si l'ús de **cerr** no és suficient per depurar el vostre codi, apreneu com utilitzar `valgrind`, `gdb` o qualsevol altra eina de depuració.
- Podeu analitzar els arxius produïts per `Game`, que descriuen com evoluciona el tauler a cada ronda.
- Conserveu una còpia de les versions antigues del vostre jugador. Feu-lo lluitar contra les seves versions anteriors per quantificar les millores.
- Assegureu-vos que el vostre programa sigui prou ràpid. El temps de CPU que es permet utilitzar és bastant curt.
- Intenteu esbrinar les estratègies dels altres jugadors observant diverses partides. D'aquesta manera, podeu intentar reaccionar als seus moviments, o fins i tot imiteu-los o milloreu-los amb el vostre propi codi.
- No espereu fins al darrer minut per enviar el jugador. Quan hi ha molts enviaments al mateix temps, el servidor triga més en executar les partides i podria ser ja massa tard!
- Podeu enviar noves versions del vostre programa en qualsevol moment.

- Recordeu: mantingueu el codi senzill, compileu-lo sovint i proveu-lo sovint, o us en penedireu.