TP3 - Juego

Dado el éxito rotundo del Tetris++ se decidió implementar un nuevo juego. A diferencia de situaciones pasadas, esta vez, el juego será elegido por los implementadores mismos. Pudiendo elegir entre tres ideas. Todas las ideas son juegos por turnos aunque muy diferentes entre si. Cada idea listada a continuación contiene las reglas de juego, los requerimientos específicos de qué se espera para la entrega, y una sugerencia de cómo atacarlo.

Nota

Para cualquiera de los juegos pueden utilizar los recursos que deseen, no es necesario que utilicen los recursos sugeridos.

Idea 1 - Spades (Bazas)

Reglas completas - Inglés (https://en.wikipedia.org/wiki/Spades_(card_game)) - Español (https://es.wikipedia.org/wiki/Juego_de_bazas)

Reglas

Comienzo de Juego

El juego de bazas puede jugarse de 2 a 4 jugadores, con naipes franceses, y consta de 13 rondas. En cada ronda, se le reparte a cada jugador una cantidad de cartas igual al número de ronda que se está jugando. En las rondas 1 a 12, luego de repartir las cartas, de develará la primer carta remanente del mazo y el palo de la misma será el palo del triunfo. En la ronda 13, el palo del triunfo es siempre Corazones.

Apuesta

Al comienzo de cada ronda, y luego de ver sus cartas, cada jugador dirá cuántas "bazas" o manos va a ganar esa ronda. La única condición es que luego de que los 4 jugadores digan cuantas "bazas" va a ganar cada uno (pudiendo decir entre 0 y la cantidad de cartas que tenga en mano) la suma entre todos los jugadores debe ser distinta al número de ronda actual. Por lo tanto, el último jugador de la ronda está limitado en la cantidad de "bazas" que dice que puede ganar. Por ejemplo: habiendo 3 jugadores en la ronda 5, si el primer jugador dice que va a ganar 1 ronda, el segundo 3, el tercero, no puede decir 1 (porque sumaría 5). Debe decir cualquier otro número entre 0 y 5.

El primer jugador de la primer ronda es elegido al azar. A partir de la segunda ronda, el primer jugador es siempre el jugador que se encontraba a la izquierda del primero en la ronda anterior (es decir, el segundo).

Jugar las Cartas

En cada mano, después de que todos los jugadores hayan anunciado su apuesta, el que juegue primero jugará una de sus cartas (la que él quiera) y los demás deberán jugar una más en el mismo orden en que anunciaron su apuesta (de izquierda a derecha), y respetando las reglas siguientes:

- Si el jugador al que le toca jugar una carta tuviera una del palo con que el jugador mano (el primero en jugar) inició la baza, deberá jugarla superando el valor de las que ya se hubieran jugado, de las de ese palo.
- Si el jugador siguiente, no tuviera una carta del mismo palo con mayor valor, pero sí una de menor valor, debe jugar esa.
- Si no tuviera ninguna carta del palo de inicio, deberá jugar un triunfo si este fuera de valor superior al de otro triunfo que hubiera echado un adversario. Si no tuviera un triunfo para superar el ya existente, pero sí tuviera de valor inferior deberá jugarlo igualmente.

- Si no tuviera un triunfo entonces podrá jugar la carta que quisiera.

El jugador que gane la mano, será quien juegue la primer carta de la siguiente.

El orden de las cartas va dado según su numeración, salvo por el 1. De manera que el 2 es el que menos vale, luego el 3, etc. hasta el 10, luego la J, la Q, la K, y luego la carta de mayor valor es el 1. De manera que: `(2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K < 1)`.

Puntaje

Al finalizar la ronda, se deben contabilizar los puntos ganados por cada jugador. Cada jugador ganará 10 puntos si adivinó el número exacto de bazas que se iba a llevar. Además, los jugadores que hubieran acertado en su previsión se llevarán 5 puntos por cada baza ganada en esa mano. Si un jugador apuestas que va a ganar 0 bazas, el mismo se llevará (además de los 10 puntos ya mencionados) 5 puntos multiplicado por el número de ronda que se haya jugado.

Ganar el Juego

Al finalizar la 12va ronda, el jugador con mayor puntaje, gana.

Requerimientos del TP

- El juego de Bazas debe poder ser jugado desde una computadora a través de una interfaz gráfica utilizando Gamelib.
- En el turno de cada jugador, se mostrarán únicamente las cartas del jugador actual (mostrando la parte de atrás de las de los demás jugadores que no les corresponda jugar).

La única diferencia con el juego real, es que al elegir las cartas para jugar cada mano, se irá de un jugador por vez en vez de todos al mismo tiempo (debido a limitaciones de mostrar todo en la misma pantalla).

Recursos

Imágenes para usar para las cartas: link (http://acbl.mybigcommerce.com/52-playing-cards/)

Posible visualización del juego: link (https://store-images.s-microsoft.com/image/apps.39246.13693784677428189.a9880959-d3d7-44ec-b7d6-14a28b4b4680.dc815216-9721-44bc-ab74-65f907738451?w=672&h=378&q=80&mode=letterbox&background=%23FFE4E4E4&format=jpg)

Sugerencia de Implementación

Se recomienda seguir una estructura *parecida* a la siguiente para el ciclo `main()`:

```
def main():
    juego = inicializar_juego()
    while not juego.terminado():
        mostrar_estado_juego(juego)
        juego.mezclar_mazo()
        juego.repartir_cartas()
        juego.pedir_apuestas()
        while not juego.ronda_terminada():
            mostrar_estado_juego(juego)
            for jugador un juego.jugadores():
                jugador.pedir_jugada()
            juego.determinar_ganador_mano()
        juego.contabilizar_puntos_ronda()
    mostrar_ganador(juego)
```

Notar que en esta implementación existe una clase `Juego` que tiene la mayoría del comportamiento. Queda a decisión del implementador decidir si utilizar esta u otras clases o no (recomendamos que usen al menos alguna clase porque les va a facilitar mantener el estado del juego).

Idea 2 - Codenames (Código Secreto)

Codenames es un juego de mesa sencillo donde dos equipos se enfrentan para descubrir a todos sus agentes en base a una serie de pistas otorgadas por el líder del equipo.

Reglas completas - Inglés (https://cdn.1j1ju.com/medias/89/5e/99-codenames-rule.pdf)

Reglas

Comienzo del Juego

Se ingresará la cantidad de jugadores y sus nombres al juego y se dividirán aleatoriamente en 2 equipos de igual cantidad de jugadores. El juego debe jugarse una cantidad N de veces siendo N la cantidad de jugadores en cada equipo. Una vez sean definidos los equipos, un jugador de cada uno de ellos será seleccionado como su Spymaster.

El juego termina luego de que todos los jugadores hayan sido el Spymaster una vez. Cada vez que se rota el rol a otro jugador se considera una ronda. El equipo ganador es aquel que sume la mayor cantidad de puntos al finalizar todas las rondas.

Cada ronda tiene una "llave" y un tablero cuadrado de 25 casilleros con una palabra en cada casilla.

Llave y Tablero

La "llave" es únicamente conocida por el Spymaster del equipo y contiene la información que vincula a cada agente con su respectiva categoría. Debido a las limitaciones de tener una única pantalla, la "llave" solo será mostrada por pantalla durante el turno de los Spymasters y, por lo tanto, el resto del equipo no debe estar mirando el tablero durante dicho periodo del juego.

El tablero contiene 25 palabras. Cada una de ellas representa a un agente que puede pertenecer a 4 categorías diferentes: Equipo azul, Equipo Rojo, Inocente o Asesino. Al final de cada ronda cada equipo suma 1 punto por cada agente que haya sido descubierto de su equipo y resta un punto por cada agente inocente que haya contactado. Cualquier equipo que contacte al Asesino pierde 5 puntos y finaliza automáticamente la ronda.

Spymaster

Durante cada ronda, un jugador de cada equipo es designado como el Spymaster. El Spymaster no puede hablar durante toda la ronda pero es capaz de ver la "llave". Su objetivo es comunicarle a su equipo quienes son los agentes que deben ser contactados en base a pistas.

Durante el turno del Spymaster, deberá revelar una pista al resto de su equipo mediante el juego. Una pista consiste en dos partes: Un número y una palabra.

La palabra de la pista debe guiar a su equipo a elegir los agentes correctos del tablero y el número corresponde a la cantidad de agentes que se relacionan a dicha palabra.

Luego de ingresar su pista, dicha pista se mostrará en el tablero de juego, se pasará al turno de los demás integrantes del equipo y una vez terminado de adivinar, se pasa al turno del siguiente Spymaster.

Jugadores

Durante cada ronda, luego de que juegue su respectivo Spymaster, los jugadores podrán intentar adivinar sus contactos. Los jugadores deben entonces clickear una palabra que ellos crean es correspondiente y en dicho caso pueden ocurrir diferentes cosas:

- Se contacta un agente del equipo de los jugadores -> Los jugadores pueden seguir adivinando y suman 1 punto.
- Se contacta un agente del equipo contrario -> El equipo contrario suma un punto y se cede el turno al Spymaster rival inmediatamente.
- Se contacta con un inocente -> Los jugadores pierden un punto y se cede el turno al Spymaster rival inmediatamente.

- Se contacta al asesino -> Los jugadores pierden 5 puntos y la ronda termina automáticamente.

En cualquier caso, los jugadores tienen un límite de intentos por turno igual a P+1 siendo P el número correspondiente a la pista otorgada por el Spymaster.

Requerimientos del TP

- Codenames debe poder ser jugado desde una computadora a través de una interfaz gráfica utilizando Gamelib.
- El tablero de palabras debe ser generado aleatoriamente en base a un archivo de palabras y debe ser mostrado en todo momento. Durante el turno del Spymaster, también debe ser visible la "llave" del juego.
- Para "contactar" un agente se le debe hacer click a la palabra del tablero que se desea revelar. Dicho casillero debe tomar el color correspondiente al agente descubierto.

Para evitar trampas al momento de generar una pista por el Spymaster debe haber un mínimo chequeo de similitud entre una palabra del tablero y una pista (es decir, si existe la palabra Mesa en el tablero, un Spymaster no puede utilizar Mesa-1 como pista). En caso de haber una detección de trampa, se deberá otorgar un agente aleatorio al otro equipo. Para una mejor detección de trampas, se sugiere utilizar SequenceMatcher de la librería difflib.

Recursos

Imagenes recomendadas para usar: link (https://puu.sh/HeHLy/68b0d728d2.zip)

Sugerencias de Implementación

Se recomienda seguir una estructura *parecida* a la siguiente para el ciclo `main()`:

```
def main():
    juego = inicializar_juego()
    while not juego.terminado():
        mostrar_estado_juego(juego)
        juego.generar_tablero()
        juego.generar_llave()
        juego.seleccionar_spymaster()
        while not juego.ronda_terminada():
            mostrar_estado_juego(juego)
            if juego.turno() == spymaster:
                juego.pedir_pista()
                if not juego.pista_es_valida():
                    juego.penalizar()
            else:
                juego.pedir_agente()
    mostrar_ganador(juego)
```

Notar que en esta implementación existe una clase `Juego` que tiene la mayoría del comportamiento. Queda a decisión del implementador decidir si utilizar esta u otras clases o no (recomendamos que usen al menos alguna clase porque les va a facilitar mantener el estado del juego).

Idea 3 - Chase (Robots)

Chase es un juego por turnos en el que el jugador debe escapar de unos robots programados para perseguir y atraparle. El jugador puede destruir los robots moviéndose de forma tal que los robots colisionen entre ellos o con otros obstáculos.

Reglas completas - Inglés (https://www.youtube.com/watch?v=K7L55s9sf4k)

Reglas

Chase se juega en una grilla rectangular bidimensional. El objetivo es escapar de unos robots que han sido programados para matarnos.

El juego es por turnos. En el juego original, el jugador comienza en una posición aleatoria. En algunas variantes (como por ejemplo GNOME Robots) el jugador comienza en el centro de la grilla. Los robots comienzan en posiciones aleatorias. Cada vez que el jugador se mueve en cualquier dirección a una casilla adyacente (horizontal, vertical o diagonal), cada robot se mueve un casillero en la dirección del jugador. Si alguno de los robots llega a la casilla del jugador, el juego se considera perdido.

Si alguno de los robots colisiona con otro robot o con un obstáculo, el robot es destruido, y en su lugar quedan sus **escombros**, que serán considerados como un obstáculo hasta que termine el nivel. (Si dos robots colisionan entre sí, quedará un solo casillero con escombros en el lugar de la colisión.)

El jugador puede empujar los escombros moviéndose de forma tal de colisionar con uno de ellos. En lugar de colisionar, el escombro se moverá un casillero en la dirección correspondiente, siempre y cuando el casillero esté vacío. En caso contrario, el movimiento no está permitido. (Esto es similar a la mecánica del juego Sokoban.)

El jugador además puede **teletransportarse** a un casillero aleatorio de la grilla. Esto cuenta como un movimiento, y los robots responderán como siempre moviéndose hacia la nueva posición del jugador. Como la posición es seleccionada aleatoriamente, es posible que el jugador se teletransporte a una casilla ocupada por un robot o unos escombros, en cuyo caso el juego se considerará perdido.

Cuando todos los robots han sido eliminados, el jugador pasa al siguiente nivel, con más robots.

Extras

No es necesario implementar los extras, pero pueden hacerlo si ya terminaron con el resto de la implementación del juego.

Algunas variantes de este juego tienen: - Una cantidad limitada (por ejemplo, una vez por nivel) de "teletransportaciones seguras", que garantizan que la casilla seleccionada está vacía. - Una cantidad limitada de usos de un arma que permite matar un robot adyacente. - Un tipo de robot que se mueve más rápido (de a dos casilleros por turno). - Un tipo de robot que no puede ser destruido (y no es necesario destruir para ganar el nivel).

Requerimientos del TP

- El juego debe ser implementado con Gamelib y tener una interfaz gráfica con la cual interactuar para jugarlo.
- El juego debe constar con al menos 3 niveles que pueden ser autogenerados dadas ciertas constantes (cantidad de robots y obstáculos que debe cambiar nivel a nivel), o estar definidos en archivos.
- El juego puede no tener final, volviendose los niveles cada vez más difícil hasta que el jugador pierda; o puede tener un final definido luego de cierta cantidad fija de niveles. Si se puede ganar, debe mostrarle al jugador de alguna manera que el mismo ganó.

Recursos

Posible visualización del juego: link (https://en.wikipedia.org/wiki/Chase_(video_game)#/media/File:Robots_graphic_screenshot.png)

Sugerencias de Implementación

Se recomienda seguir una estructura *parecida* a la siguiente para el ciclo `main()`:

```
def main():
    juego = inicializar_juego()
    while not juego.terminado():
        juego.inicializar_siguiente_nivel()
        while not juego.nivel_terminado():
            mostrar_estado_juego(juego)
            accion = pedir_accion_al_jugador()
            juego.avanzar_un_step(accion)
    if juego.ganado():
        mostrar_pantalla_ganador()
    else:
        mostrar_pantalla_perdedor()
```

Notar que en esta implementación existe una clase `Juego` que tiene la mayoría del comportamiento. Queda a decisión del implementador decidir si utilizar esta u otras clases o no (recomendamos que usen al menos alguna clase porque les va a facilitar mantener el estado del juego).