Bitácora Programación 3

Día 1: Decisión de rasgos generales del proyecto,

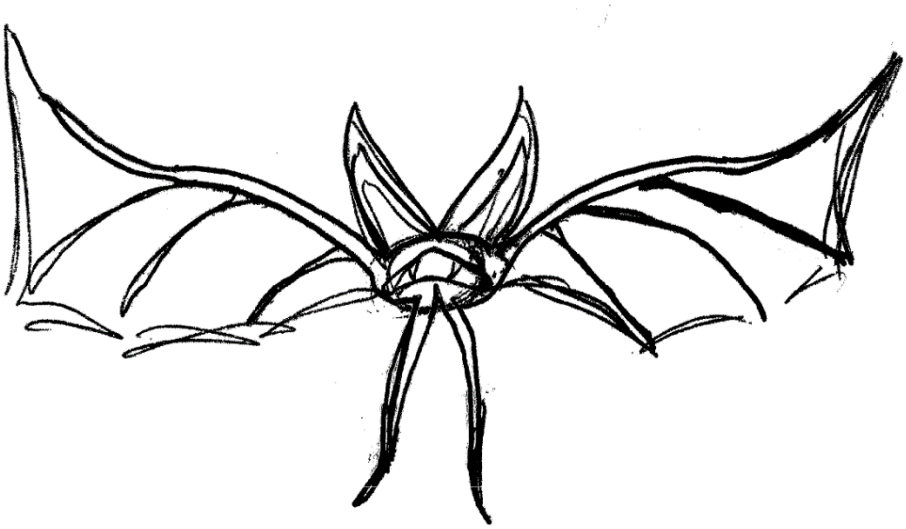
- El objetivo del proyecto es crear un juego multiplataforma con *orientación a las plataformas portables* (Android, IOS, WP).
- El juego sería diseñado en 2D por temas de complejidad de diseño estético, pero se intentará abstraerlo de esta limitación para hacer posible un eventual cambio al 3D.
- Se decidió utilizar Unity3D como herramienta básica para realizar el proyecto (acordamos lenguaje C# para homogeneizar).
- Para simplificar la comunicación remota y el control de versiones del proyecto, *decidimos gestionar utilizando* GitHub.
- Decidimos orden de desarrollo del proyecto (ver adjuntos).
- Decidimos temática del juego (ver adjuntos).
- Las herramientas utilizadas seran registradas en un adjunto.
- Realizamos *pequeña* prueba de las herramientas, realizando un control de movimiento básico.

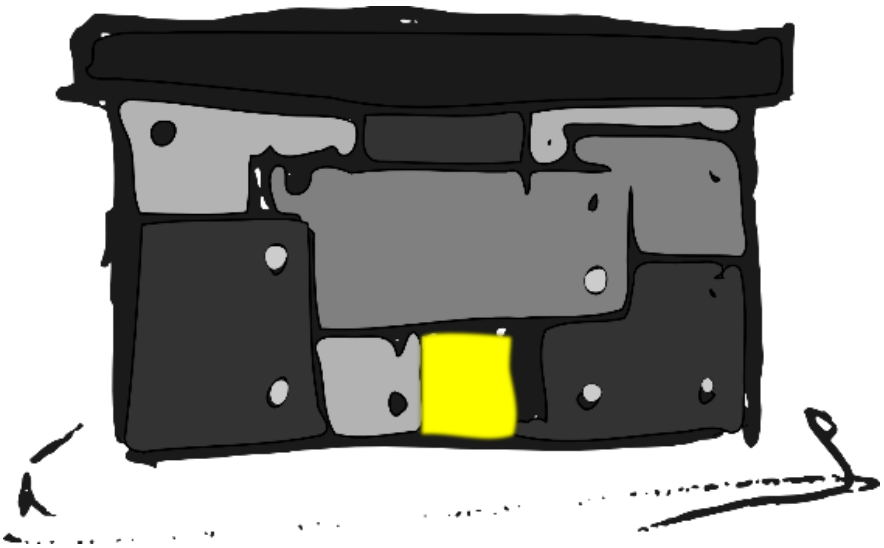
Dia 2: Bocetos código,

- Testeamos distintas escenas, cargar recursos entre ellas y planteamos un boceto muy básico de como funcionaría el Save/Load y la persistencia entre escenas del juego.
- *Planteamiento rudimentario de interacciones* (con mouse y colisiones).
- Agregado randomizado de color (pelo y gorra), para luego probar Save-Load.
- Decidimos utilizar Unity Remote para facilitar el debugging en sistemas Android.
- *Deberíamos decidir próximamente los datos necesarios para guardar* una partida, y con ello la forma de *guardarlo (¿es necesaria una base de datos elaborada?)*
- Posibles temas inmediatos a desarrollar:
    - Sistema *Save/Load*.
    - Creación de *juego nuevo*.
    - Definición de *clases* necesarias (*Especie-Monstruo* por ejemplo).
    - *Sistema de batalla*.

Día 3: Bocetos de estilo,

- Bocetamos en lapiz y papel distintos dibujos *concept-art* del juego, con el objetivo de decidir cual encaja mejor y cual nos resulta más cómodo de hacer a lo largo del proyecto.
- Intentamos *limpiar* los diseños hechos, de forma digital.

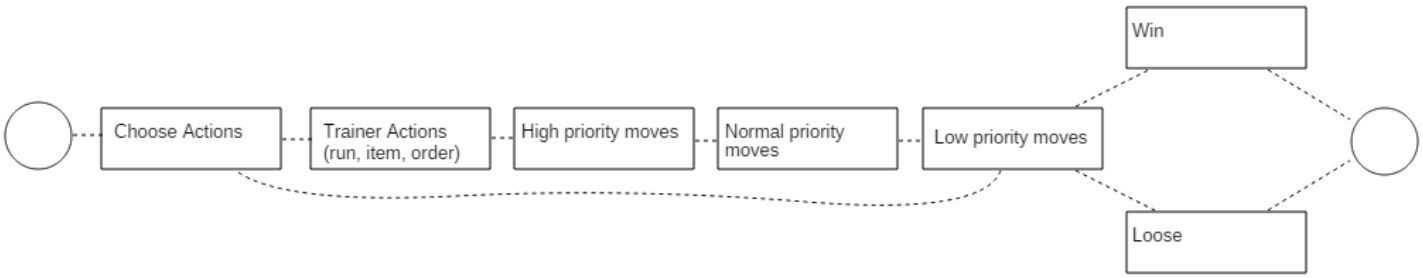


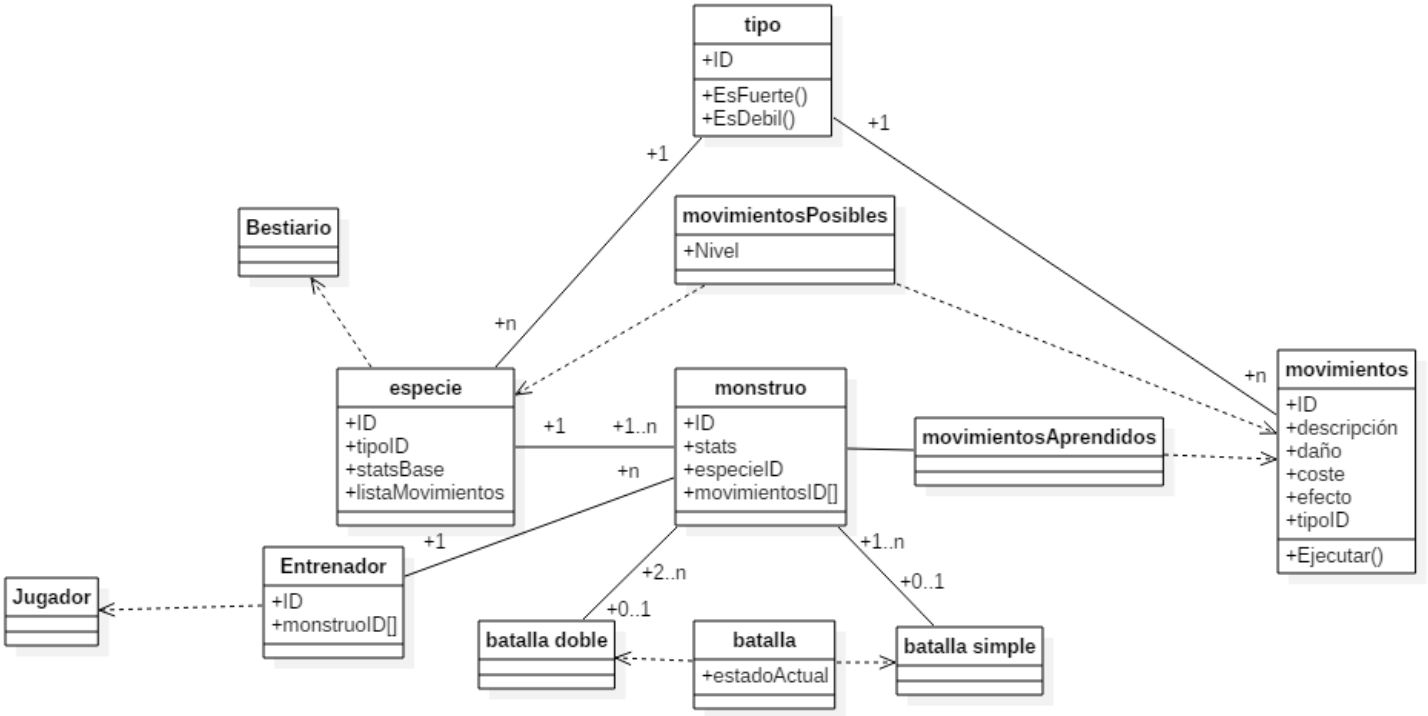


Coloreados con InkScape

Día 4: Bocetos UML,

- Planteamos los primeros bocetos de ***transición de estado de las batallas***, y de ***clases.***

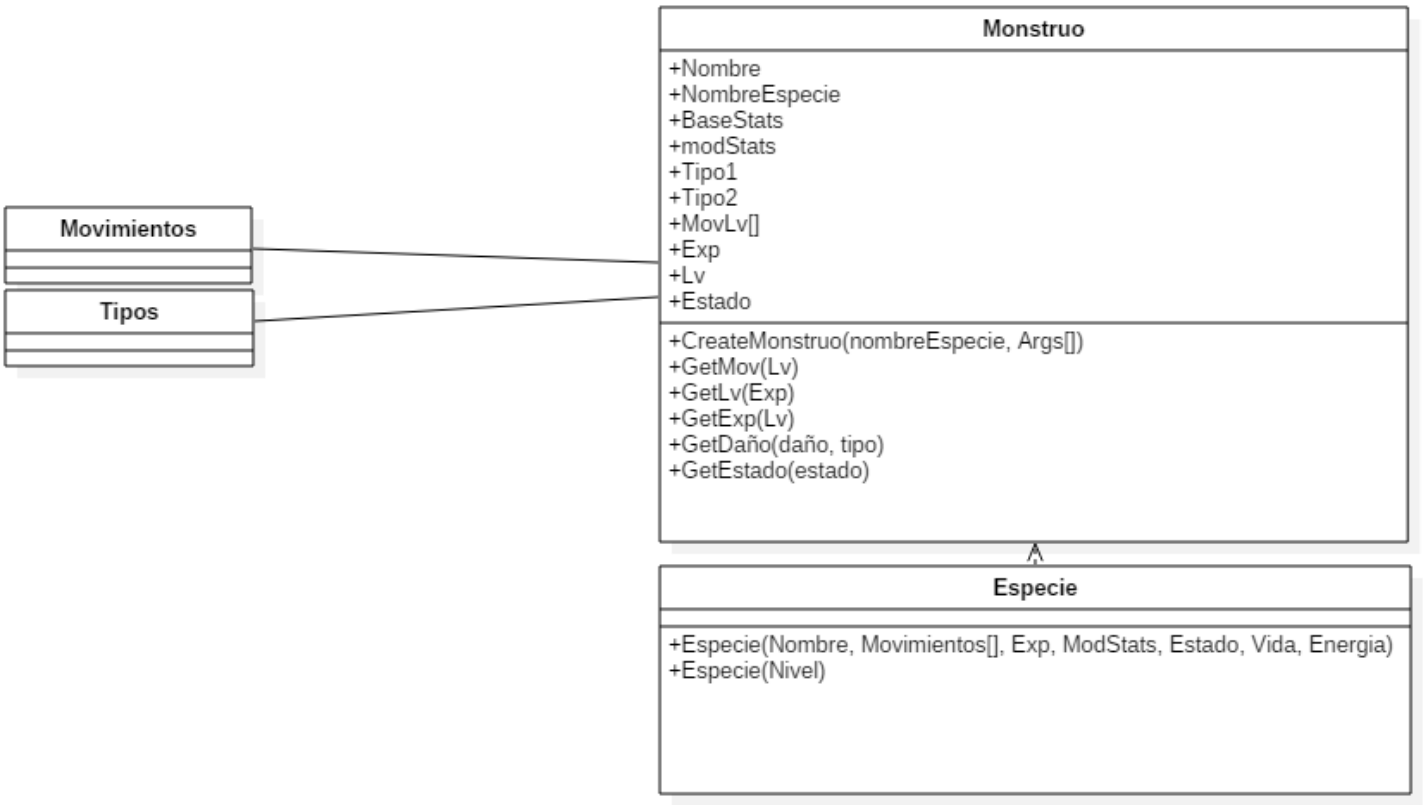




- Avances en la escena de batalla (Funcionalidad *básica* de acciones y prioridades de movimientos).
- Agregados cambios de escena para testear funciones (huir y entrar a batalla).
- *Agregado Save/Load* utilizando como datos los colores de pelo y gorra.
- ***Save/Load ahora maneja posición del jugador*** (reflejado al salir/entrar del juego o al salir de un combate).
- Pronto habría que implementar un *new/continue* en la pantalla de inicio.

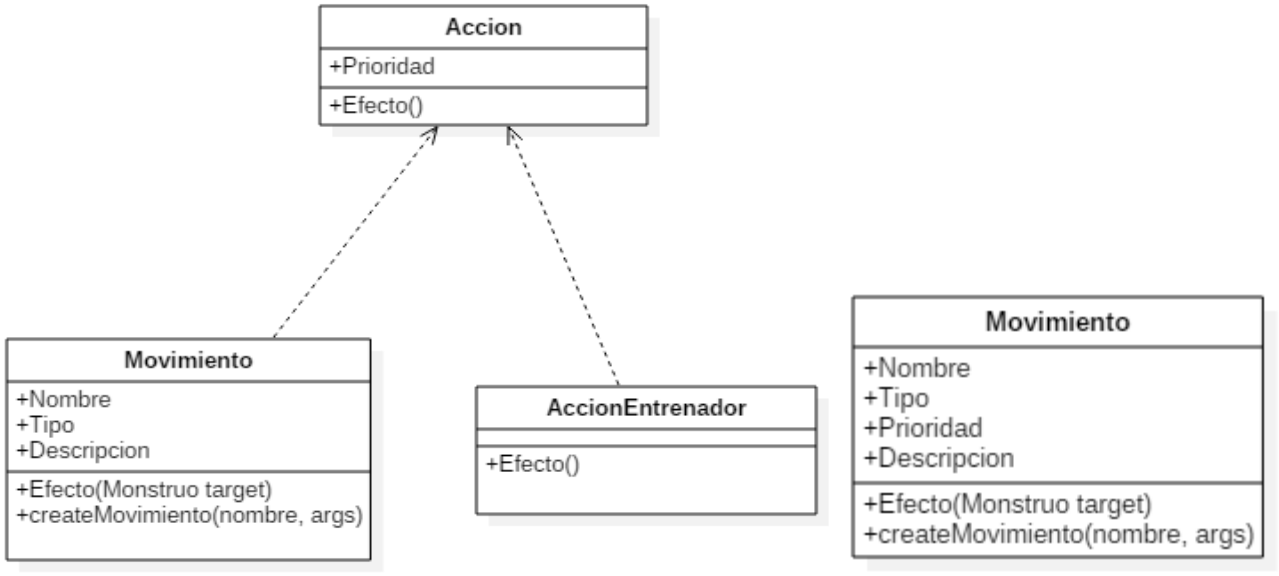
Día 5: Cambio total de clases,

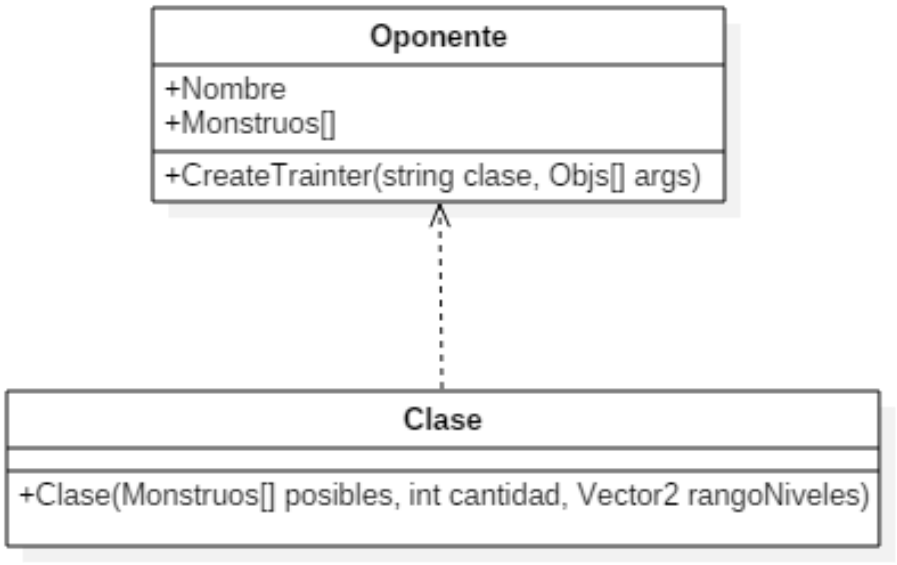
- Fueron modificadas completamente las clases de *monstruos* (y *especies heredadas*), acciones y batalla, pensando ya en respetar los nuevos UML y en el Save/Load de los mismos.



Día 6: Planes y UML's

- Nos reunimos para discutir como continuar con el desarrollo del proyecto, y aprovechamos la ocación para ***continuar con el desarrollo de algunos diagramas***.
- Decidimos ***que se van a utilizar bases de datos*** para manejar los datos guardados de las partidas (y asi facilitar la posibilidad de manejar ***multiples partidas***).





- Próximamente deberíamos trabajar en los DER para diseñar las bases de datos que implementaremos.

Día 7: Foco en sistema de batalla,

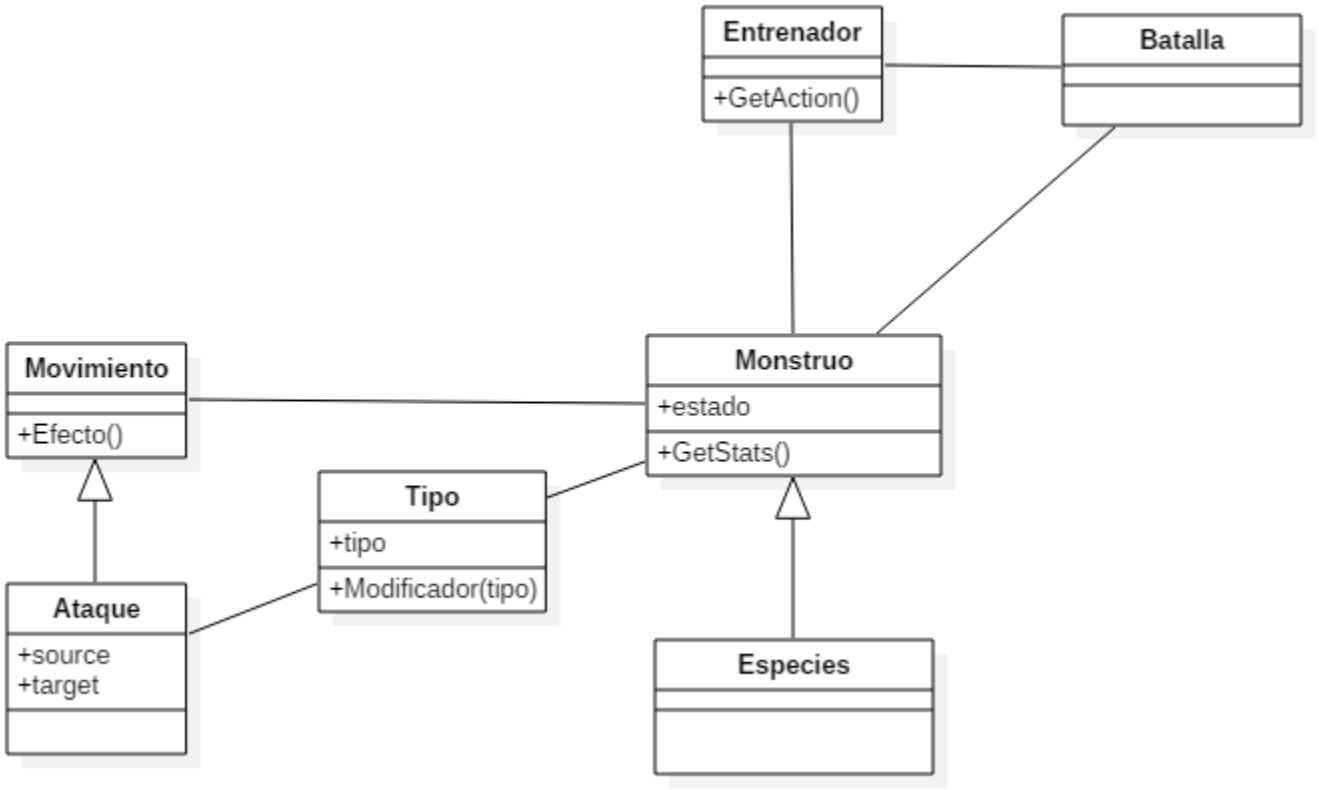
- Reestructuramos las *acciones*, tal y como con *Monstruos*, ahora todas las acciones heredan de una clase *Accion* abstracta, con un ***método fábrica*** que llamará a cada accion ("Huir", "ItemX", "CambiarX", "AtaqueX"). En el estadio *elegir* se espera a que se designe una acción, y luego la efectuara en su turno correspondiente. Probablemente sea necesario agregar un parámetro al constructor, designando un *target* (a tener en cuenta).
- Creadas clases de *Entrenador*, *acciones* y *batalla* reestructurado para encajarle.
- Creado menu de elección de ataques (el de items deberia ser similar).

Día 8: Foco en sistema de batalla,

- Avanzamos en el *sistema de batalla*, hasta el momento se puede ***elegir el tipo de acción*** (atacar, cambiar monstruo, huir), y dentro de esos tipos de acción ***se puede especificar la acción a ejecutar*** (el ataque, cual monstruo insertar).
- Hay por el momento dos movimientos *ataque* creados, en los que se puede observar como el movimiento se relaciona con los monstruos en batalla.
- Se creó una primera *inteligencia artificial* básica del oponente, en la que elige un ataque al azar para ejecutar contra el jugador.
- Se implementaron barras de vida para mostrar la salud de los monstruos en batalla.

Día 9: Foco en sistema de batalla,

- Leves cambios en batalla. Incluyen modificaciones en apariencia y randomizado de oponentes (diferenciado por clase de oponente).
- Próximos pasos a seguir para avanzar en la batalla deberían ser Captura de monstruos y tipos de ataque/monstruos (con fortalezas y debilidades).
- *Implementados tipos* (Monstruos y movimientos rediseñados para utilizarlos).



UML de la batalla hasta el momento.

Día 10: Randomizados y cargados,

- Se agregó ***randomizado de monstruos por habitat.***
- Se modificó la batalla para ***cargar al oponente al comienzo.***
- Se planteó un *catchRate* para posibilitar eventualmente *Atrapar* al monstruo oponente.

Día 11: Menú principal y experiencia,

- Se creó un *menú principal* (opciones nueva partida y contunuar).
- Se implementó la experiencia de los monstruos (y los niveles en relación a la misma).

Día 12-13: Bases de datos,

- Implementamos una *Base de datos* básica para el juego.
- Despúes de muchisimo renegar, logramos hacerla correr en Android.
- Próximo paso va a ser completar la implementación de la base de datos para con eso dar por finalizado el esqueleto del juego.

Día 14-16: Puliendo,

- La base de datos efectua correctamente las transacciones que involucran posicion de jugador.
- Dentro de la partida, *persisten los datos de monstruos* correctamente (estado actual, experiencia, etc).
- Se puede *atrapar monstruos salvajes* correctamente.
- Se solucionaron varios detalles ***menores*** (warnings de compilacion, codigo *legacy*, etc.

Día 17: Base de datos de Monstruos,

- Se terminó de implementar la *base de datos para los monstruos*.