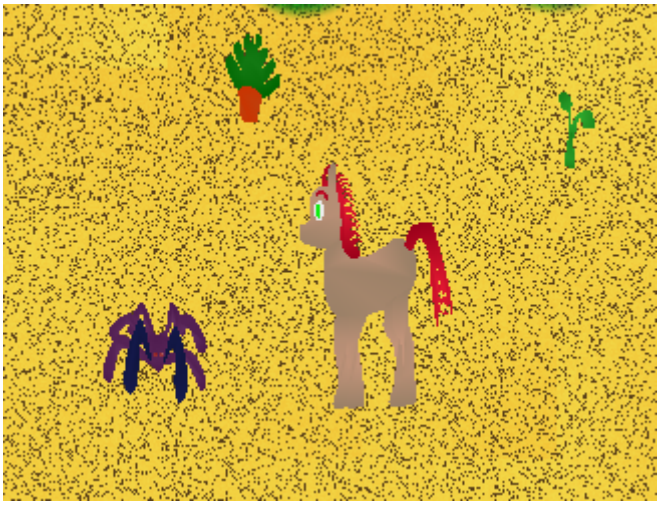
Hooves And Fangs



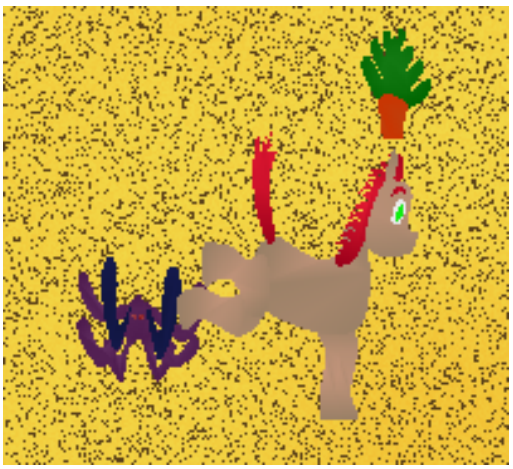
(Hooves si dirige a proteggere una carota ed un germoglio)

Proteggete il vostro orto!

In Hooves and Fangs scenderete nei panni di **Hooves**, un pony che vive una vita tranquilla in campagna, e coltiva buonissime carote.

Un giorno però il clan dei F.A.N.G.S. (Field-Assault Nesting Ground Spiders), formato da temutissimi aracnidi, ha deciso di saccheggiare il vostro orto delle carote che avete coltivato con fatica!

Non resta altro che mettere i vostri ferri di cavallo migliori e cacciarli via!



(Hooves che calcia un fang)

Gameplay

Comanderete il pony e potrete muovervi per tutta la grandezza del campo coltivato.
Periodicamente compariranno sia germogli che fang nel campo.
I germogli vanno difesi fino a quando crescono in carote, le quali possono essere raccolte semplicemente toccandole per aumentare il punteggio.
I fang impiegano del tempo per decidere se attaccare il pony o la carota più vecchia. Dopo aver deciso andranno dritti verso il loro obiettivo.
Se un fang tocca il pony o una carota attaccherà e distruggerà la carota oppure farà del danno al pony.
Il pony può scalciare i fang dietro di sé uccidendoli. I fang deceduti ritorneranno alla madre terra dopo qualche secondo.
Quando il pony finisce i punti vita è Game Over.

Comandi

Tastiera e Mouse.

Tasto sinistro del mouse - seleziona gli elementi della UI.

W - muove il pony verso l'alto.
A - muove il pony verso sinistra.
S - muove il pony verso il basso.
D - muove il pony verso destra.

SPAZIO - il pony scalcia dietro di sé.

ESC - Mette il gioco in pausa.

Punteggio

Ogni carota raccolta il punteggio aumenterà di 10 punti.
Una volta conclusa una partita, quando il pony perde tutta la vita, è possibile salvare il proprio punteggio assegnandogli un identificativo di 3 lettere.
È possibile consultare i punteggi precedenti nella sezione SCOREBOARD del menù, e si possono eventualmente resettare tutti i punteggi.

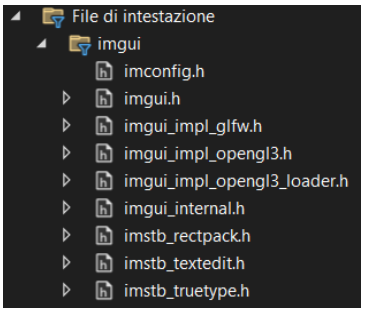
Come Giocare a Hooves and Fangs

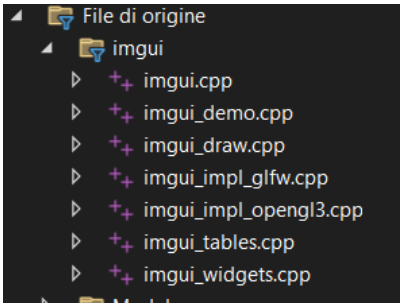
Guida per poter scaricare e giocare Hooves and Fangs:

1) Scaricate i file da questo link

2) Estraete i file.

3) Aprire un progetto su Visual Studio e aggiungere le dipendenze di imgui, opengl (incluso glm), glfw, glad.c e tutto quello presente in dependencies_GL_GLFW.
In Progetto -> Proprietà -> C/C++ -> "*Directory di inclusione aggiuntive*" aggiungere *dependencies_GL_GLFW* e *dependencies_GL_GLFW/include*.
In C/C++ -> Preprocessore aggiungere **_CRT_SECURE_NO_WARNINGS** per poter utilizzare fscanf. In Linker -> Generale -> "*Directory librerie aggiuntive*" aggiungere *dependencies_GL_GLDW/lib*.
In Linker -> Input -> "Dipendenze aggiuntive" aggiungere *glfw3.lib* e *opengl32.lib*.
Aggiungere i file header e cpp presenti in dependencies_GL_GLFW/ImGui in due filtri del progetto di nome imgui.





4) Adesso che ci sono le dipendenze aggiungere tutti i file di progetto. Assicurarsi che i *.txt* e gli shader *.glsl* siano nella stessa cartella del progetto.

5) Compilate e avviate (**F5**).

Figure fatte con curve di Hermite

Per questo progetto, per le carote, i semi, i fang e il pony, vengono utilizzate delle curve di hermite.

Ogni figura ha un numero di curve indicizzato, e due colori assegnati ad ogni curva.

Le curve nomefigura.txt vengono generate attraverso un programma esterno (Editor Curve), e poi viene affiancato un altro file nomefigura_color.txt per l'assegnazione dei colori per ogni curva.

Animazioni

Le animazioni sono state fatte in due modi:

1) Variando la matrice di modellazione di ogni figura basica o per ogni curva di una figura fatta con curve di Hermite.
2) Riassegnando una curva specifica in una figura fatta con curve di Hermite.

Le animazioni del tipo 1 comprendono:

- il cambio di direzione del pony: quando il pony cambia direzione da destra verso sinistra o viceversa, l'intera figura di hermite gradualmente ruota rispetto all'asse y.
- il movimento delle zampe e della testa del pony, il movimento delle zampe dei fang e delle loro fauci durante gli attacchi, il movimento oscillatorio dei semi e delle foglie delle carote: attraverso dei calcoli a base sinusoidale (periodico) modifico la matrice di modellazione solo di alcune curve specifiche e/o solo in determinate condizioni (ad esempio il pony muove le zampe solo mentre si sta muovendo), ruotandole rispetto all'asse z nel sistema di riferimento della figura intera.
- Le zampe posteriori del pony: vengono scalate orizzontalmente.

Le animazioni del tipo:

- invece di ricreare / caricare un'intera figura con tutte le sue curve, riassegno solo le curve che mi interessa muovere, avendo il beneficio di modificare la forma di una curva senza influire troppo negativamente sulle performance.

Gli attacchi del pony e dei fang utilizzano entrambi i metodi:
Vengono riassegnate alcune delle curve (le zampe posteriori del pony e le zanne dei fang) che poi vengono animate attraverso delle modifiche della matrice di modellazione.

Shader GLSL

Ci sono due shader presenti nel programma. Uno è lo shader per il campo di gioco, che contiene la *grassland* e il *field* dove si muovono i personaggi e crescono le carote.

Da [thebooksofshaders] ho preso le funzioni *noise* e *random* per lo pseudorandom.

Per il campo interno calcolo prima il bordo simil seghettato a 0.85 di distanza dal centro. Nel programma il field del model (la parte di logica del programma, dove si muovono i personaggi) è rimappato in -0.8 __ 0.8, quindi a 0.85 comprendo tutto il campo.

```
  // Coordinate ricentrate in -1 _ 1
  vec2 centered = abs(uv * 2 - 1);
  // Border of the field
  float edgeNoise = noise(uv * 15.0) * 0.02; // Rough rounded shapes
  // Tiny sawtooth pattern
  float sawToothX = abs(sin(uv.x * 80.0)) * 0.005;
  float sawToothY = abs(sin(uv.y * 80.0)) * 0.005;
  // The distance from the center, modified by noise
  float dist = max(centered.x, centered.y) + edgeNoise + sawToothX +
sawToothY;
  // Smooth Border at distance 0.9 from center
  float fieldBorder = smoothstep(0.855, 0.845, dist);
```

Sia per lo shader del field che per quello della grassland oscillo prima col vento in entrambi gli assi, poi calcolo attraverso noise e random se un pixel fa parte dei cespugli più scuri o sono erba normale. Poi oscillo il colore dell'erba nel tempo sinusoidalmente per dare l'effetto del vento su una pianura.

```
  // Wind oscillation
  float windX = sin(uv.y * 5.0 + iTime * 2.0) * 0.02;
  float windY = cos(uv.x * 5.0 + iTime * 1.5) * 0.02;
  vec2 grassUV = uv + vec2(windX, windY);

  float grassGrain = random(grassUV * 250.0);
  float grassPatches = noise(grassUV * 40.0);

  vec3 baseGrass = mix(darkGrass, grassGreen, grassGrain);
  vec3 grassAreaColor = (grassPatches < 0.15) ? bushColor : baseGrass;

  // Add movement to grass
  grassAreaColor += vec3(0.05) * sin(uv.x * 20.0 + iTime * 3.0);
  // Mix the two zones based on the rough noisy border mask
  vec3 finalColor = mix(grassAreaColor, fieldColor, fieldBorder);
```