Hello Hackers!!

Inside this folder we have included some of the data stream from our game *Sky: Children of the Light*. It is player data taken from the first level of our game *Isle of Dawn*.

Picture of the first are of the game, Isle of Dawn

If you have any questions, advice, want swag, or just want to hang out please come visit us at our booth

Challenge

Description: Sky: Children of the Light has an amazing and vibrant community of socially minded players, and we at thatgamecompany work tirelessly to ensure that players feel safe and supported. To that end, we encourage and help foster relationships in our game through reinforcement of positive behaviors in a compassionate and supporting environment.

For this challenge, we've provided a set of 3D player telemetry data, including the time and position of movements, chat messages, instruments played, hands held, items purchased, etc. Given this data, we challenge participants to design and implement analyses and data visualizations that identify meaningful relationships within our game. For example, sentiment analysis can be used to determine the emotional tone of exchanges from chat data. We are also interested in maintaining the spirit of play in our game. For this challenge, participants can also opt to focus on identifying behaviors that attempt to compromise the integrity of our game systems, such as egregious gameplay hacks or attempted abuse of the underlying technology. ### Criteria

Projects will be judged based on the following criteria:

- Effectiveness of analysis
- Novelty of approach
- Novelty of conclusions
- Clarity of presentation (including visuals used to convey results)

Prizes

Placement	Prize
1st	Choice of Steam Deck, Nintendo Switch, or Amazon Gift card
2nd	Amazon gift card(s)
3rd	Amazon gift card(s)

Data Insights

Here is a little information about the data structure.

Schema:

```
type EventName =
  | "ping"
  | "hand_held"
  | "chat_msg"
  | "wingbuff_collect"
  | "wingbuff_drop"
  | "candle_purchased"
  | "spirit_shop_item_purchased"
  | "got_wax"
  | "healed_player";
type SkyEvent = {
  time: number;
  userId: number;
  platform: "android" | "iphoneos" | "nx" | "huawei";
  pos: [number, number, number];
  fps: number;
  event: Record<EventName, number>;
};
```

Key Descriptions

Event Name	Description
time	millisecond timestamp when the event was recorded
userId	unique identifier given to each user
platform	the type of device used to play the game. We are cross platform. `nx` is nintendo switch
country	country code for the ip address the user is using
events	hash map of events that have occurred within the last 5 seconds for that user.

Event Name Descriptions

Event Name	Description
ping	Pings are generated whenever the two conditions happen first: minimum distance of 10 units or a minimum time of 30 seconds between pings. Pings will only be sent when the app is in the front.
hand_held	When a player holds the hand of another player or NPC - not triggered by the handhold leader, only followers
chat_msg	When a chat message is sent by the local player
wingbuff_collect	When a wing buff is picked up (shiny boy)

Event Name	Description
wingbuff_drop	drop wing buff due to damage
candle_purchased	triggered by in app purchase
spirit_shop_item_purchase	item purchase with ingame currency
got_wax	in game currency pickup
healed_player	when the player the user chose to heal is healed to the point of standing up

Mini example of aggregating some of the data:

```
const fs = require("fs");
const readline = require("readline");

const readStream = fs.createReadStream("dawn-event-data.json", "utf-8");
const rl = readline.createInterface({ input: readStream });

async function loadDataset() {
  let skyEventCount = 0;
  const eventTypeCountAggregation = {};
  for await (const line of rl) {
    if (line.startsWith("[") || line.startsWith("]")) continue;
    const cleanedLine = line.trim().replace(/,$/, "");
    const data = JSON.parse(cleanedLine);
    skyEventCount += 1;
    for (const [key, count] of Object.entries(data.events)) {
      if (eventTypeCountAggregation[key] === undefined)
        eventTypeCountAggregation[key] = 0;
      eventTypeCountAggregation[key] += count;
    }
  }
  return { skyEventCount, eventTypeCountAggregation };
}

loadDataset().then((data) => {
  console.log(`read everything. result is ${JSON.stringify(data, null, 2)}`);
});
```