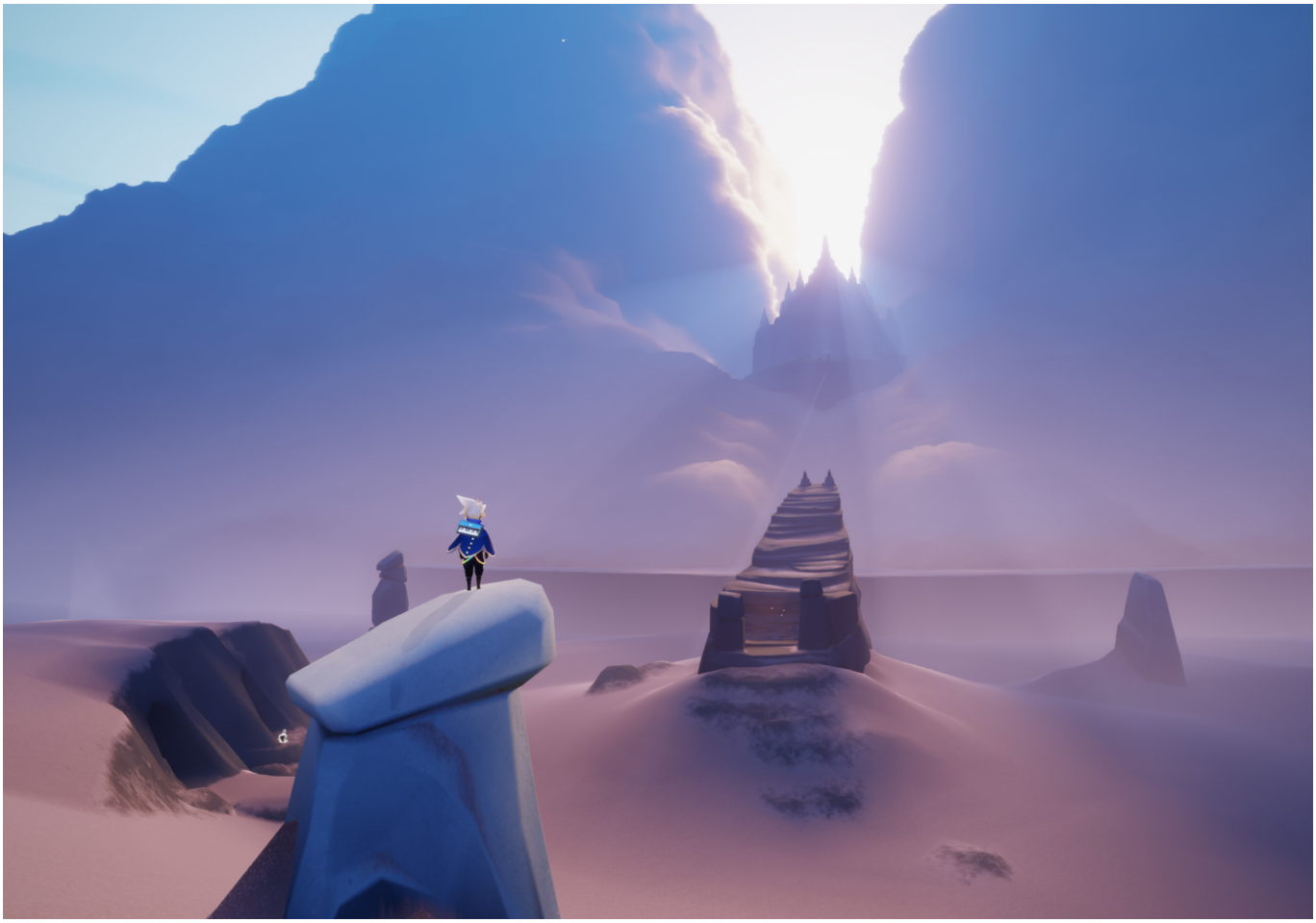
Hello Hackers!!

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*.



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

Challenge

Sky has an amazing and vibrant community of socially minded players, and we are tirelessly working to ensure that players feel safe and supported. As with any online system, we are not immune to bad actors attempting to undermine our game ecosystem and platform. While we do not request identifiable information from our players and we make significant efforts to protect their privacy, we do track certain events in Sky in order to improve our player experience. 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. Our challenge is to come up with a novel way to use, interpret, or visualize this data, potentially in the service of identifying player relationships, threats to player safety (negative behavior, harassment), or other bad actors (such as hackers or bots), but participants should feel free to take it in any direction they think is most interesting.

Criteria

Projects will be judged based on the following questions:

- How novel is your solution?
- What sort of insights does your solution enable?
- How could your hack help support the community?

Prizes

Placement	Prize
1st	Nintendo Switch(s), Amazon gift card(s)
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)
wingbuff_drop	drop wing buff due to damage
candle_purchased	triggered by in app purchase
spirit_shop_item_purchased	purchase with in game 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)}`);
});
```