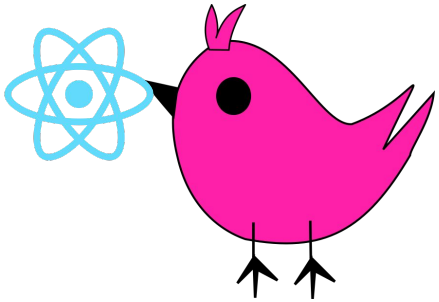
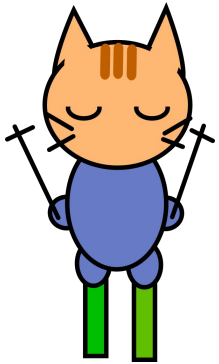


My Strange Addiction: React Games



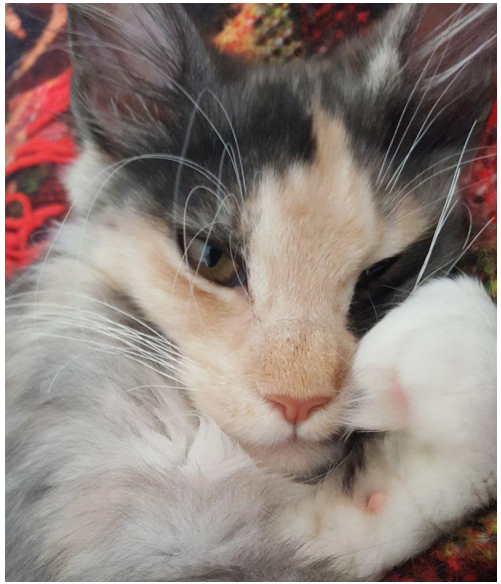
Cerize Santos

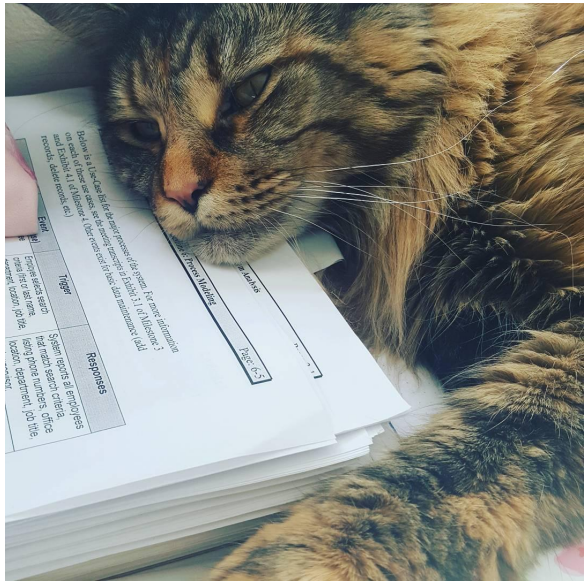
ReactJS Vancouver, Jan 24 2018

About me

Software developer at bananatag

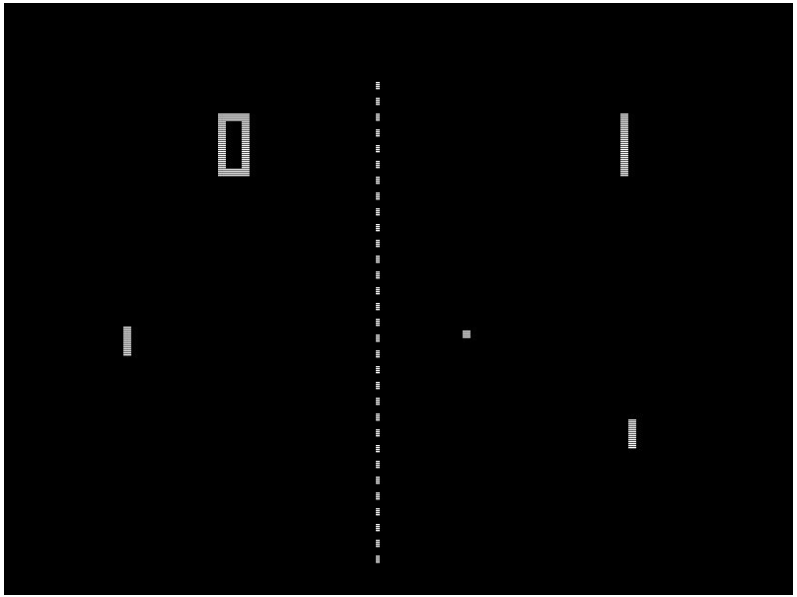
Work with **all things Javascript** - Node/React/Redux





Video Games

Pong, 70's



Anno 1800, 2018

Credits: Ubisoft

Making your own toys

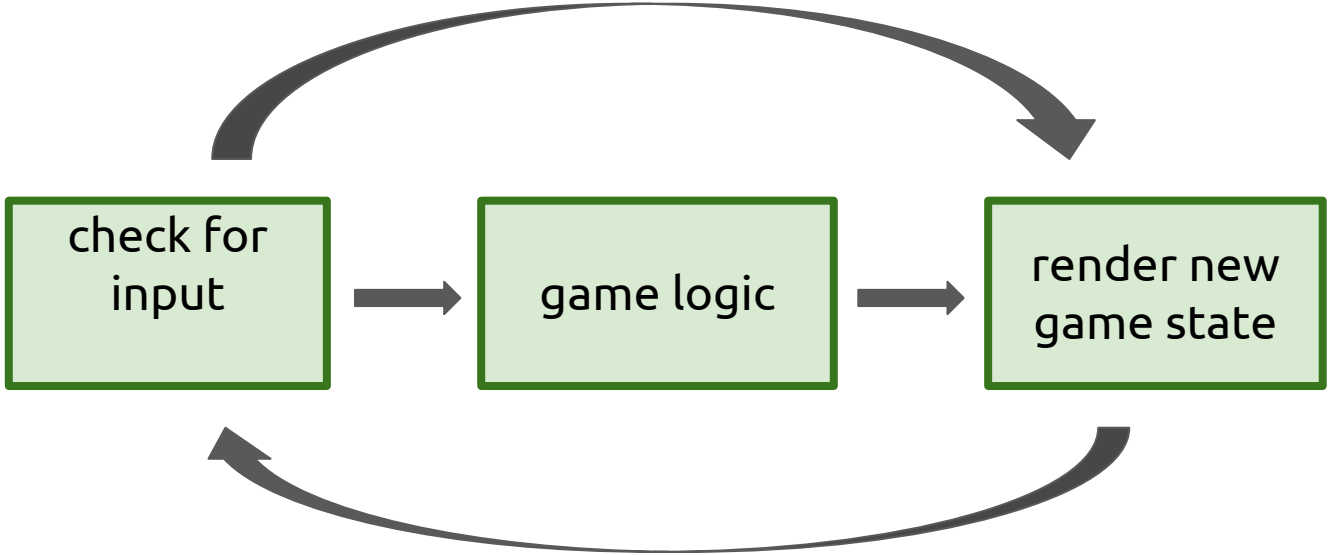
devgames.io

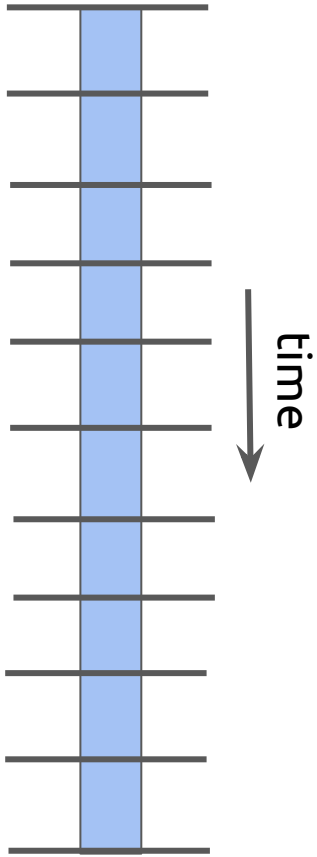
A few tips

1.

Understand the loop pattern

The Loop pattern

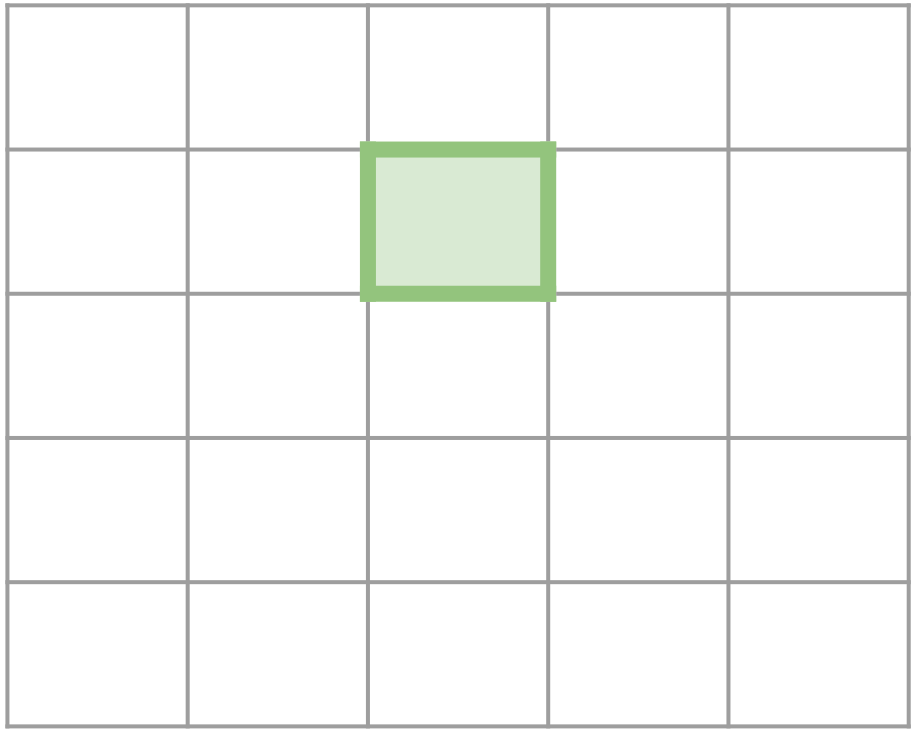




related to → FPS

2.

Whenever possible, use a matrix



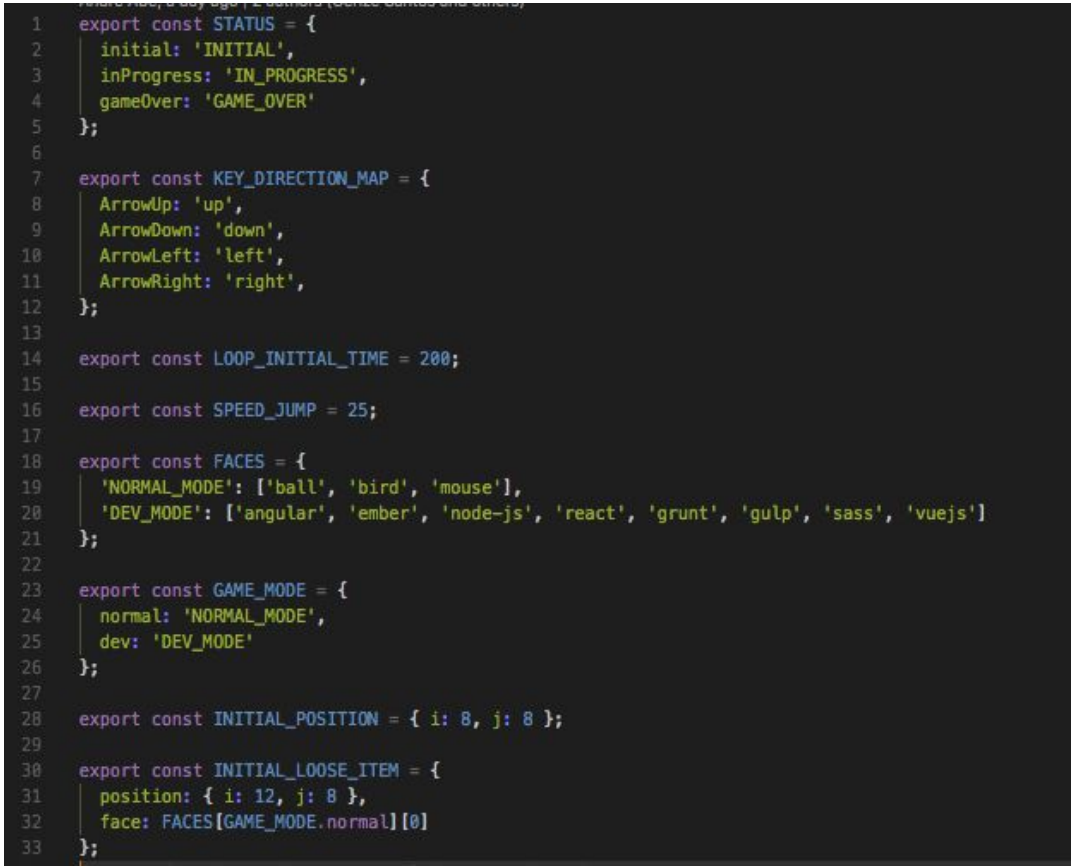
How many states?

How will they be represented?

Will they be local React state in a global store?

3.

Use constants, make your code flexible



```
export const STATUS = {
  initial: 'INITIAL',
  inProgress: 'IN_PROGRESS',
  gameOver: 'GAME_OVER'
};

export const KEY_DIRECTION_MAP = {
  ArrowUp: 'up',
  ArrowDown: 'down',
  ArrowLeft: 'left',
  ArrowRight: 'right',
};

export const LOOP_INITIAL_TIME = 200;

export const SPEED_JUMP = 25;

export const FACES = {
  'NORMAL_MODE': ['ball', 'bird', 'mouse'],
  'DEV_MODE': ['angular', 'ember', 'node-js', 'react', 'grunt', 'gulp', 'sass', 'vuejs']
};

export const GAME_MODE = {
  normal: 'NORMAL_MODE',
  dev: 'DEV_MODE'
};

export const INITIAL_POSITION = { i: 8, j: 8 };

export const INITIAL_LOOSE_ITEM = {
  position: { i: 12, j: 8 },
  face: FACES[GAME_MODE.normal][0]
};
```

```
const getInitialBoard = (boardSize) => {
    const squares = {};
    // Get the number of bombs in the board - 20% of the squares
    const nBombs = Math.floor(boardSize * boardSize * 0.2);
    // Assign positions to the bombs
    const bombPositions = _getBombsPosition(nBombs, boardSize);

    for (let i = 1; i <= boardSize; i += 1) {
        for (let j = 1; j <= boardSize; j += 1) {
            const position = `${i}${j}`;
            const question = getRandomQuestion();
            squares[position] = {
                position,
                status: SQUARE_STATUS.closed,
                face: _getFace(position, boardSize, bombPositions),
                question: question.title,
                answer: question.answer
            };
        }
    }

    return {
        state: SQUARE_STATUS.initial,
        nBombs,
        squares
    };
};
```

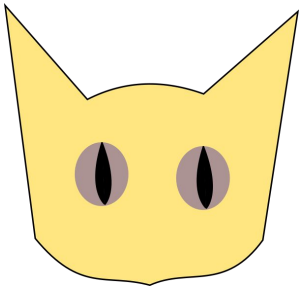
4.

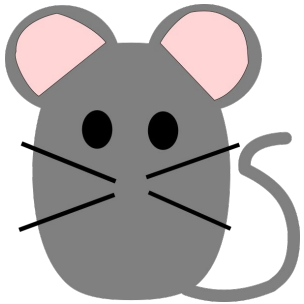
Find that one piece of logic that makes it all work

```
function _moveBody(head, shouldAddNewItem) {
  const { body, looseItem } = store;

  const newBody = body.map((part, index) => {
    // First body part occupies the position of the current head
    if (index === 0) {
      return { position: _.clone(head), face: part.face, gameMode: part.gameMode };
    }

    // Other body parts occupy the position of anterior parts
    return ({ position: _.clone(body[index - 1].position), face: part.face, gameMode: part.gameMode });
  });

  if (shouldAddNewItem) {
    // new item occupies the position of the last item of the current body
    const newItem = {
      position: (body[-1] && _.clone(body[-1].position)) || head,
      face: looseItem.face,
      gameMode: looseItem.gameMode
    };

    newBody.push(newItem);
    _createLooseItem();

    // Make it harder and start new loop
    if (newBody.length % 10 === 0) {
      clearInterval(intervalId);
      speed -= SPEED_JUMP;
      _move();
    }
  }

  store.body = newBody;
}
```

5.

Don’t get stuck because of illustrations

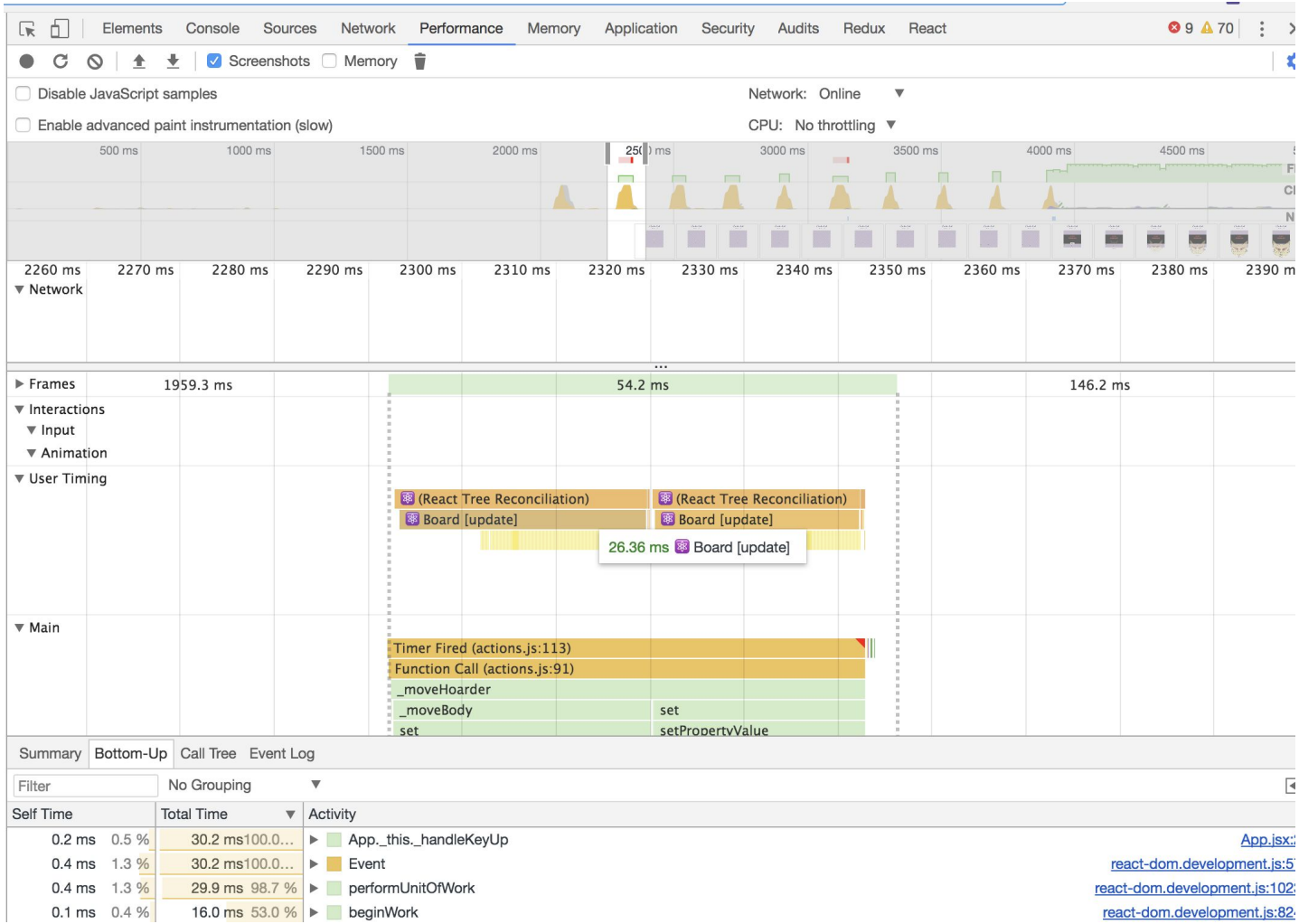


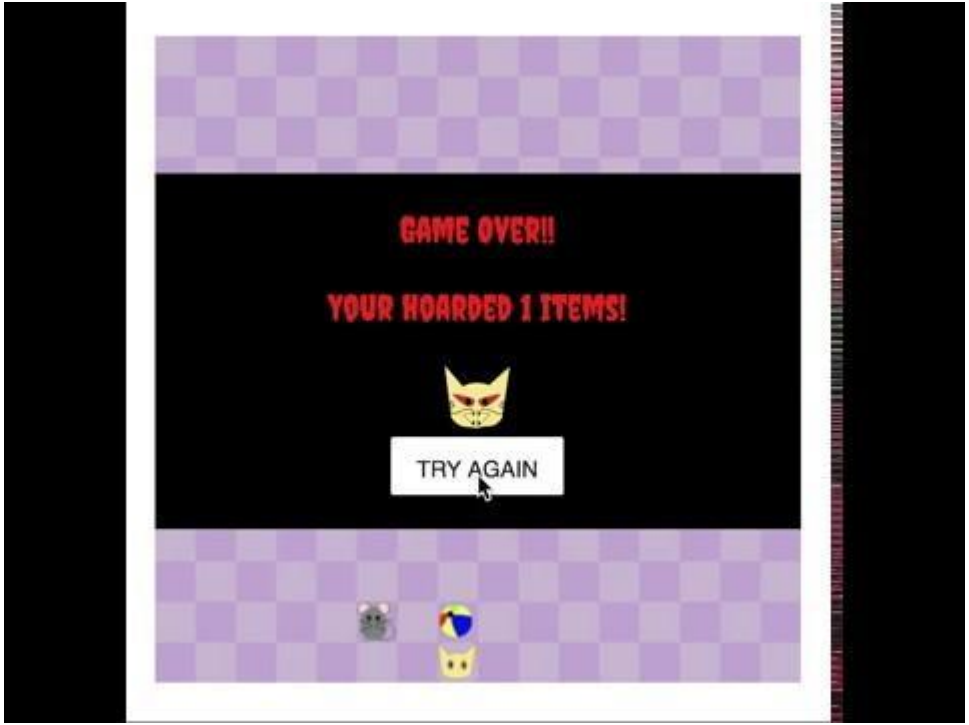


6.

Test edge cases that can affect performance ASAP

React 16
+
dev mode
+
Chrome Dev Tools




GAME OVER!!
YOUR HOARDED 1 ITEMS!
TRY AGAIN

7.

Establish your data pattern early on

8.

Keep your mess compartmentalized

```
if (action.type === 'UPDATE_OBSTACLES_POSITION') {
  let speed = action.payload.speed;
  let direction = action.payload.direction;
  let score = 1;
  if (state.gameStatus === 'TEMP_PAUSED') {
    return state;
  };
  let obstacles = [];
  let skierPosition = state.skierPosition;
  let newGameStatus = state.gameStatus;
  let newScore = state.score + score;
  let newSpeed = 3;

  state.obstacles.forEach((obstable) => {
    if (obstable.yPosition + action.payload.speed < 500) {
      const obstacleInfo = {
        yPosition: obstable.yPosition + speed,
        xPosition: obstable.xPosition + (direction * -2)
      };
      if (!ignoreObstacle.includes(obstable.key) && hitTest(skier, {
          left: obstacleInfo.xPosition,
          bottom: obstacleInfo.yPosition,
          width: obstable.type.width,
          height: obstable.type.height
        })) {
        skierPosition = 'p6';
        ignoreObstacle.push(obstable.key);
        newGameStatus = 'TEMP_PAUSED';
        newScore -= 100;
      }
      obstacleInfo.visible = (obstable.xPosition >= -obstable.type.width && obstable.xPosition <= 500 + obstable.type.width);
      obstacles.push(Object.assign({}, obstable, obstacleInfo))
    } else {
      const index = ignoreObstacle.indexOf(obstable.key);
      if (index >= 0) {
        ignoreObstacle.splice(index, 1);
      }
```

9.

Establish what you want from the project

- **Improve your React skills?**
  - Follow all best practices and patterns
- **Test a new library/tool?**
  - Use it as much as possible, even if it doesn't need it
- **Make a game to be successful?**
  - Original idea
  - Beautiful graphics
  - Is React the right tool?

Thank you

Andre Abe (abe.andre@gmail.com)

Bananatag Team

Get in touch

cerizesantos@gmail.com