Bugs em jogos e suas aplicações em speedruns

Gustavo Spelzon Caparica
Supervisores: Prof. Dr. Alfredo Goldman, Guilherme Amantea e Wilson Kazuo Mizutani
Instituto de Matemática e Estatística - Universidade de São Paulo

Qual a relação entre *speedruns* e *bugs*?

- *Speedrunning* é a prática de terminar jogos ou fases o mais rápido possível, um *hobby* competitivo de uma grande comunidade online de jogadores.
- Muitas *speedruns* utilizam falhas de programação dos jogos para pular partes do jogo e economizar tempo de diversas maneiras. Para realizar isso do melhor jeito possível, a comunidade analisa e estuda como *bugs* nesses jogos funcionam.
- Esse processo é bem parecido com *engenharia reversa*, pois os jogadores não tem acesso ao código fonte original, e precisam analisar o comportamento do jogo como uma caixa preta.
- Mesmo em jogos muito diferentes, existem características recorrentes referentes aos tipo de *bugs* descobertos e utilizados por *speedrunners*.
- Conhecer esses aspectos facilita o trabalho de descoberta de novos *bugs* por parte de *speedrunners*.
- O uso de ferramentas especializadas para analisar o estado do jogo enquanto roda também é bastante útil para entender o comportamento da implementação de jogos.

Tipos de *bugs* tipicamente úteis para *speedruns*

- Quando presentes, esses tipos de *bugs* geralmente podem ser utilizados por *speedruns* para economizar tempo de alguma maneira.
- Escapar dos limites do nível (*Out of Bounds*)
  - Permitem acessar áreas do jogo antes do normal, pulando seções ou níveis inteiros. Pode ser consequência de problemas com a verificação de colisão ou com a limitação da velocidade máxima do jogador.

Figura 1: Acessando posições fora dos limites do nível, é possível acessar áreas antes do normal. *Super Mario 64*

- Tratamento incorreto em certos casos de borda
  - Problemas clássicos como *overflow* de inteiros ou *buffers* sob certas circunstâncias podem ser usados para manipular a memória do jogo de maneira benéfica ao jogador, por exemplo duplicando itens ou modificando vida de inimigos.
  - As vezes o código usado para modificar a velocidade do jogador possui problemas em certas situações que podem ser utilizados para se movimentar muito mais rapidamente do que o possível normalmente.
- Sobreposição ou interrupção de eventos dentro do jogo
  - Ao ativar dois eventos simultaneamente de um jeito que não deveria ser possível, o estado do jogo fica inconsistente de alguma maneira. Pode significar que eventos longos possam ser interrompidos por eventos curtos, minimizando o tempo que o jogador precisa esperar. Dependendo da implementação do jogo e do tipo de evento interrompido, pode levar a um estado de jogo muito diferente do esperado, o que por sua vez pode levar a descoberta de outros *bugs*.
- Corrupção de memória ou do fluxo de execução do programa
  - Se for possível utilizar um *bug* para corromper valores importantes do jogo, por exemplo o número de elementos de algumas lista, pode ser possível utilizar aspectos da implementação para modificar valores de memória diretamente. Dependendo do tipo de manipulação possível, pode ser possível manipular o jogo a executar um trecho qualquer de código, definido pelo jogador.

Figura 2: Ao corromper o número de itens de seu inventário, o jogo exibe valores de memória como itens com nomes e valores estranhos, que podem ser diretamente manipulados. *Pokémon Red*

Exemplo: Forçar o jogo a carregar seções de mapa incorretamente

- Nos jogos da quarta geração de *Pokémon* (*Diamond* e *Pearl*), os mapas do jogo são implementados por meio de seções quadradas de tamanho **32** por **32** de modo que, a cada momento, apenas **4** seções estão carregadas na memória do jogo e visíveis. A medida que o jogador se movimenta no mapa o jogo transparentemente carrega as informações de novos setores adjacentes antes que eles sejam visíveis ao mesmo tempo que descarrega os dados de setores que não são mais visíveis.

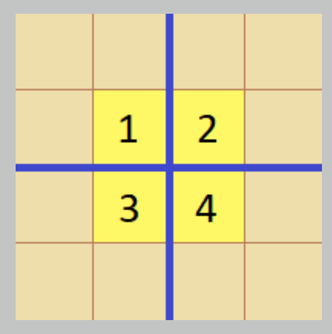

Figura 3: Ilustração das quatro subseções. ao atravessar uma das linhas azuis, o jogo carrega uma nova seção do mapa.

- Ao usar a bicicleta, um item obtido ao longo do jogo, é possível se movimentar rápido o suficiente para que esse sistema funcione incorretamente, resultando em seções carregadas em posições incorretas (por exemplo uma seção que normalmente fica ao norte carregada na posição oeste do jogador) ou carregadas parcialmente, por exemplo sem gráficos ou *NPCs*. Se o jogador continuar andando em uma direção sem realizar esse *bug* novamente, o jogo carrega as próximas seções corretamente e similarmente, se o jogador abrir certas páginas do menu como a *Pokédex* o jogo recarrega as seções corretamente quando o menu é fechado, o que torna possível controlar os efeitos dessa corrupção.

Figura 4: Exemplo de possíveis resultados ao realizar esse *bug*.

- Esse *bug* pode ser usado para passar por áreas que normalmente bloqueiam o jogador, sobrescrevendo essas áreas com informação de uma seções adjacentes sob as quais o jogador consegue passar. Outra aplicação é se movimentar na área *Out of Bounds* de uma maneira específica, possibilitando o acesso a mapas inacessíveis, ou ir diretamente para o final do jogo.

Tool-assisted speedrunning

- Uma *Tool-assisted speedrun* (*TAS*) é uma *speedrun* criada com a ajuda de emuladores com funcionalidades extras.
- Do ponto de vista do código do jogo, existe na verdade um valor para cada botão do controle e o jogo repetidamente coleta todos os valores em um determinado instante de tempo, com um intervalo curto entre cada leitura para garantir responsividade.
- Um *TAS* é uma sequência desses valores, que ao ser reproduzida por um emulador determinístico sempre leva ao mesmo resultado. A aleatoriedade dentro do jogo depende apenas do *input* do jogador ao longo do tempo.
- O objetivo de um *TAS* é mostrar como seria a *speedrun* perfeita, desconsiderando o "fator humano", para fins de entretenimento. Para atingir esse objetivo, é muito comum abusar falhas de programação nos jogos.
- O foco em entretenimento existe para motivar a criação de *TASes* surpreendentes, pois o uso de ferramentas torna certas coisas previsíveis e entediantes, como por exemplo nunca ser atingido por inimigos, já que o *TAS* não vai realizar um erro.

Comunidades *online* relacionadas a *speedrunning*

- `https://speedrun.com/` Possui placares e recursos para um grande número de jogos
- `https://tasvideos.org/` Comunidade de *TASers* com muitos recursos relacionados a emuladores e *bugs* em jogos
- `https://sourceruns.org/` *Speedrunners* focados em jogos de primeira pessoa da engine *Source*