GameSnacks Developer Guide

(external)

For questions, please contact: gamesnacks-devs@google.com

Last updated: 2/24/2021

GameSnacks Game Requirements

Core Gameplay Only

GameSnacks games should be a continuous experience of core gameplay. The **following should be removed** from GameSnacks games:

- Advertising and sponsorships
- Sharing prompts (e.g. "Click to tweet your score.")

Snackable Games

GameSnacks games should be

- Easy to learn: a wide range of users should be able to learn how to play in less than 30 seconds. Games that require some domain knowledge (e.g., a chess tactics game) are acceptable. Include in-game instructions and onboarding where appropriate.
- Fast-paced: the user should be able to earn a score within 15 seconds.
- Without text: aside from the instructions, the game itself should not rely on text that the user has to read in order to succeed.
    - If there is any text, the text should be in English.
    - When possible, use symbols and visual tutorials instead of text. For example: show a finger making an action instead of using text to describe the action.

Using the gamesnacks.js API

We provide an JavaScript API that games use to check audio status, report scores, and report gameOver events. It's called "gamesnacks.js" Include it with:

```
<script
src="https://embed.gamesnacks.com/assets/js/gamesnacks.js"></script>
```

This is a small script that allows games to plug in to the GameSnacks API. We recommend including it after your HTML, but before your other javascript.

API Tester

We provide an API Tester tool which allows developers to verify if their games are correctly integrating with GameSnacks APIs. It can be used to verify locally-running games. It's located at https://gamesnacks.com/demo/test.

Score API

GameSnacks keeps track of score data for high scores and other features. **When the user's score changes**, you should use the sendScore function in the API like this:

```
// var score = users current score
GAMESNACKS.sendScore(score)
```

For games that traditionally don't show scores, e.g. puzzle games, please use this API to send an incrementing score upon the completion of each level. There is no need to show this score in the game UI.

GameOver API

GameSnacks also uses "game over" events for competitions, ads and other features. It should be triggered on failure outcomes, e.g. when the user dies or was unable to complete a level. Game Over events should also be triggered when the user restarts, the user exits a level, the user clicks the home button to go back to the main menu, or other similar events that cause the session to end. You should use the gameOver function in the API like this:

```
GAMESNACKS.gameOver()
```

LevelComplete API

For games with levels, triggered when a level is successfully completed by the user. Pass in a number representing the index of the level in the overall level ordering. Levels should be indexed starting from 1.

- Even if your game doesn't have traditional "levels", call this API if your game has discreet ordered success milestones that are natural stopping points. For example: in this balls game, call levelComplete on every successful user fling.
- If your levels are unordered, call this api with a level of 0. E.g. in this fishing game, call GAMESNACKS.levelComplete(0) at the end of every fishing round.
- Some games don't have levels which is okay, e.g. this reversi game.

Use the levelComplete API like this:

```
// var level = index of level completed by user
GAMESNACKS.levelComplete(level)
```

Audio API

GameSnacks games should use this audio API so that GameSnacks can provide consistent volume controls. There are two ways to integrate with the audio API: a getter method and a PubSub method.

```
// Getter method. Returns a boolean of whether audio is enabled.
GAMESNACKS.isAudioEnabled()

// PubSub method. Accepts a callback function that will be called
// with a boolean value of the new value of isAudioEnabled whenever
// it changes.
GAMESNACKS.subscribeToAudioUpdates(callback);
```

Audio getter method
If using the getter method, you can call the getter at any time to see if audio is enabled. You should check before each sound effect.

```
if (GAMESNACKS.isAudioEnabled()) {
  playSoundEffect();
}
```

Audio PubSub method
Optionally, instead of checking the getter method before each sound effect, you can check it once at the beginning of the game, and then subscribe to changes. This is suitable for things like background music.

```
if (GAMESNACKS.isAudioEnabled()) {
  playBackgroundMusic();
}
GAMESNACKS.subscribeToAudioUpdates((isAudioEnabled) => {
  if (isAudioEnabled) {
    playBackgroundMusic();
  } else {
    stopPlayingBackgroundMusic();
  }
});
```

GameReady API

GameSnacks games should call gameReady() as soon as the main menu (*NOT* loading screen) is both displayed and interactive. This may, for instance, be used to hide GameSnacks loading screens or indicators. At this point the game, or at least the first level, should be loaded.

```
GAMESNACKS.gameReady()
```

Rewarded Ads APIs

Implement this section if creating rewarded ads games on GameSnacks

Rewards must not have value outside of your app, they must not have (or be easily exchanged for) monetary value, and must not be saleable or exchangeable for goods and services, and you should not encourage accidental clicks on ads or reward prompts.

GAMESNACKS.rewardedAdOpportunity

Steps

1. When the user reaches a point in the game with reward potential (common examples are main menu and end-game screen), call rewardedAdOpportunity() to notify GameSnacks. Pass in callbacks to trigger game behavior as it relates to the ad.
2. If conditions are met, GameSnacks will call beforeReward(), at which point the game should render UI to the user indicating they can watch an ad for a reward. Don't render rewarded UI before this, as there may not be an ad available.
3. If the user clicks into rewarded ad UI, call showAdFn(). Always call the most recent showAdFn. Previous showAdFn callbacks are invalidated on every beforeReward call.
4. GameSnacks will call beforeBreak() shortly after, at which point the game should pause and mute while the ad plays.
5. adComplete() or adDismissed() will be called next, depending on whether the user watched the entire ad. If adComplete() is called, give the user a reward.
6. afterBreak() will always be called regardless of if the user watched the ad to completion. Here you should unpause and unmute the game.

```
GAMESNACKS.rewardedAdOpportunity({
    beforeReward,
    beforeBreak,
    adComplete,
    adDismissed,
    afterBreak
  }: {
    // Show reward UI. Call showAdFn() if clicked.
    beforeReward: (showAdFn: () => any) => any,
    // Mute and pause the game.
    beforeBreak: () => any,
     // Player watched the ad to completion. Give them the reward.
    adComplete: () => any,
    // Player dismissed the ad before it finished.
    adDismissed: () => any,
    // Unmute and unpause the game.
    afterBreak: () => any,
  });
```

Sandboxed iframe restrictions

GameSnacks games should function in a sandboxed iframe with the following properties. The API Tester will verify this.

```
<iframe
  sandbox="allow-scripts allow-forms allow-same-origin"
  src="<your-game-url>"
></iframe>
```

GameSnacks games should not use web features restricted by the above iframe sandbox parameters, including:

- Modals such as window.alert()
- Popups
- Top-level window navigation
- The Pointer Lock API

Saving Progress

For games with level progression, long-term progress, collectables, etc. you should use localStorage to save the state of the game. When the user opens the game for subsequent sessions, it should load any saved progress. It is ok if the game state isn't *exactly* the same (e.g. characters don't have to be in the exact position), but if the user gets to level 6, the game should start the user in level 6 again.

Examples

- Saving unlocked skins, characters
- Saving unlocked levels
- Saving collectables
- Saving high score (if high score is in the UI)
- Saving other in-game achievements

Examples of games that do not require saving progress

- https://gamesnacks.com/embed/games/tower_v2
- https://gamesnacks.com/embed/games/timberguy
- https://gamesnacks.com/embed/games/cakesliceninja

No external scripts or servers

GameSnacks games should not rely on anything (no images, assets, sounds, scripts, etc.) that is outside of the zip file (other than gamesnacks.js). In addition, you may not make calls to any external servers for any purpose. If your game needs an external server, please contact us for next steps.

Compatibility

Games should function well in the environments listed below. Adding platform-specific control schemes is encouraged (e.g., ability to use arrow keys on desktop.) Games will be shown in portrait orientation. Landscape support is not required.

- Mobile: iOS and Android. Chrome, Firefox, and Safari.
- Desktop: Chrome, Firefox, and Safari.

Size and Efficiency

GameSnacks games are played in countries with slower internet, so smaller game bundle sizes are important! GameSnacks games should be as lightweight as possible. Please compress all images and minify scripts to reduce bundle size. If possible, defer loading of assets needed for later levels until they are needed.

Performance

GameSnacks games run on a wide variety of devices, and so to maintain a great user experience, we require that they can still play well on lower end devices. Specifically:

During gameplay*, 95% of frames must run at 30 fps or faster (i.e. under 33.3 ms/frame) on a Samsung Galaxy J2 equivalent phone.

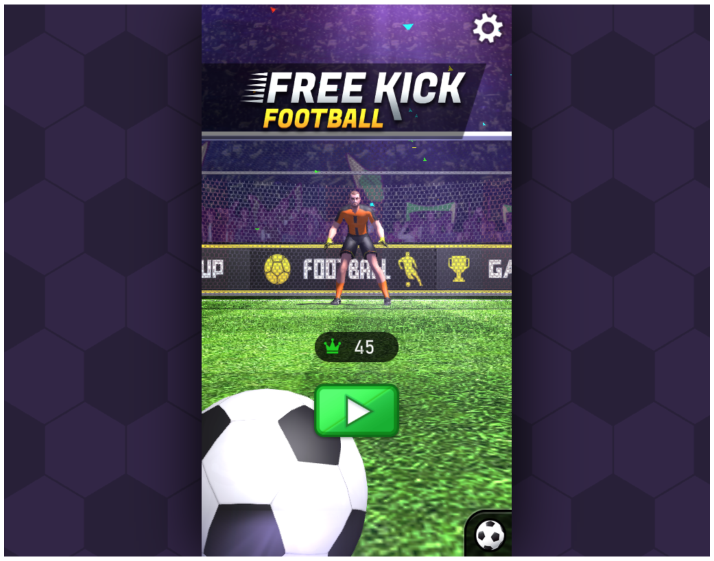
* It is OK if the framerate is lower on menus and loading screens

Splash page for the game (added 2/12/2021)

Each game should have a game specific splash page that appears while the game is loading. This can be a static screen or include an animated spinning wheel or loading bar. The splash should include the logo for the game or use the game's app icon (1:1 ratio). The Splash page goes away automatically when the game has loaded.

Side tiles for portrait-only orientation (added 2/12/2021)

Games that are locked in portrait are ok as along as they have a tiled background that shows on desktop. The objective is to add an intentional look to the sides of the game rather than black or grey. The tiled background can be used for the loading screen if desired.
Side bar tile tips:

- File size - Keep file size under 10K or smaller.
- Dimensions - The height and width of the tile can vary, as long as the tile repeats and the size is under 10K.
- Graphics - The tile should support the graphics of the game keeping in mind, the more complex the design the larger the file may be.
- Color - Color the tile with a darker shade found in the game. Also, select a base color for unde the tile for slow-loading pages.
- Example from Free Kick with side tile:
- 

Assets

The following are the minimum set of assets to include when submitting a game to GameSnacks.

- Please see the GameSnacks Thumbnail Guide before creating assets. Assets should follow these guidelines.
- **App icon**
    - Needs to work at all sizes, sometimes as small as 60x60. Looking for an iconic representation of the game. Do not include logos in app icons.

        - **[   ]** 1:1 App icon 512x512
- **Thumbnails**
    - Includes a logo, character(s), game parts, etc
        - **[   ]** 4:3 Wide 1, 1024x768
        - **[   ]** 16:9 Wide 2, 1024x576
        - **[   ]** 2:3 Tall, 683x1024
- **Layers**
    - We ask that you provide the artwork for your “4:3 Wide 1, 1024x768” thumbnail size as a composite image AND as separated files with transparent backgrounds. Most thumbs can be separated into 3 layers, although sometimes a 4th layer is needed. The logo should be included in one of the layers.
        - **[   ]** Foreground objects on transparent
        - **[   ]** Logo on it’s own layer on transparent
        - **[   ]** Midground objects on transparent
        - **[   ]** Background flat with a solid fill
- *An example of all delivered asset files:*
    - *gamename-app-icon.png*
    - *gamename-thumb-wide1.png*
    - *gamename-thumb-wide2.png*
    - *gamename-thumb-tall.png*
    - *gamename-layer-fore.png*
    - *gamename-layer-logo.png*
    - *gamename-layer-mid.png*
    - *gamename-layer-bg.png*

File Structure

When submitting a game to GameSnacks, the root directory should contain the following:

- index.html: the root html document that loads the game.
- Game assets

File Delivery

First, convert the game directory into a ZIP file. Submit the ZIP file via email to gamesnacks-devs@google.com and rachaelzb@google.com