Drop-down Terminal

Introduction

A drop-down terminal pops up from the top of the screen in video game console fashion and can be toggled with a single hotkey. Applications such as Yakuake, Guake or Tilda provide drop-down terminal functionality for the regular desktop environments. With awesome and the power of lua, however, we can mimic this functionality and still use our precious light-weight terminal applications.

Adding the following function to your rc.lua and calling it in a keybinding will create a new window for the drop-down terminal when it does not exist, and will toggle between hidden and visible if one does exist. The first argument is the program to run (eg. “urxvtc”), the second argument is the height (absolute pixels when > 1 or a height percentage when < 1, 0.2 (20% of the screen height) by default), and the third argument is the screen to toggle on. The second and third arguments are optional.

Function

```
-- This function is for awesome versions prior to 3.4

dropdown = {}

function dropdown_toggle(prog, height, s)
   if s == nil then s = mouse.screen end
   if height == nil then height = 0.2 end
   if not dropdown[prog] then
      -- Create table
      dropdown[prog] = {}
      -- Add unmanage hook for dropdown programs
      awful.hooks.unmanage.register(function (c)
                                       for scr, cl in pairs(dropdown[prog])
do
                                          if cl == c then
                                             dropdown[prog][scr] = nil
                                          end
                                       end
                                    end)
   end
   if not dropdown[prog][s] then
      spawnw = function (c)
                  -- Store client
                  dropdown[prog][s] = c
                  -- Float client
                  awful.client.floating.set(c, true)
                  -- Get screen geometry
                  screengeom = screen[s].workarea
                  -- Calculate height
```

```
                if height < 1 then
                   height = screengeom.height*height
                end

                -- I like a different border with for the popup window
                -- So I don't confuse it with terminals in the layout
                bw = 2

                -- Resize client
                c:geometry({
                             x = screengeom.x,
                             y = screengeom.y - 1000,
                             width = screengeom.width - bw,
                             height = height - bw
                          })

                -- Mark terminal as ontop
                --            c.ontop = true
                --            c.above = true
                c.border_width = bw

                -- Focus and raise client
                c:raise()
                client.focus = c

                -- Remove hook
                awful.hooks.manage.unregister(spawnw)
             end

   -- Add hook
   awful.hooks.manage.register(spawnw)

   -- Spawn program
   awful.util.spawn(prog)

   dropdown.currtag = awful.tag.selected(s)
else
   -- Get client
   c = dropdown[prog][s]
   -- Switch the client to the current workspace

   -- Focus and raise if not hidden
   if c.hidden then
      awful.client.movetotag(awful.tag.selected(s), c)
      c.hidden = false
      c:raise()
      client.focus = c
   else
      if awful.tag.selected(s) == dropdown.currtag then
         c.hidden = true
         local ctags = c:tags()
```

```
            for i, t in pairs(ctags) do
               ctags[i] = nil
            end
            c:tags(ctags)
         else
            awful.client.movetotag(awful.tag.selected(s), c)
            c:raise()
            client.focus = c
         end
      end
      dropdown.currtag = awful.tag.selected(s)
   end
end
```

Another solution

The previous solution have two little quirks: the console window is detected as the first window being managed just after requesting one. There could be a race condition but it is unlikely. The second associated quirk is that when you restart awesome, you lose the fact that this window is a drop-down terminal. You now have one terminal which is sticky. This is a bit disturbing. Here is another solution that works around this by relying on a name given to the dropdown terminal:

```
-- Quake like console on top
-- Similar to:
--   http://git.sysphere.org/awesome-configs/tree/scratch/drop.lua

-- But uses a different implementation. The main difference is that we
-- are able to detect the Quake console from its name
-- (QuakeConsoleNeedsUniqueName by default).

-- Use:

-- local quake = require("quake")
-- local quakeconsole = {}
-- for s = 1, screen.count() do
--    quakeconsole[s] = quake({ terminal = config.terminal,
--                  height = 0.3,
--                              screen = s })
-- end

-- config.keys.global = awful.util.table.join(
--    config.keys.global,
--    awful.key({ modkey }, "`",
--       function () quakeconsole[mouse.screen]:toggle() end)

-- If you have a rule like "awful.client.setslave" for your terminals,
-- ensure you use an exception for
-- QuakeConsoleNeedsUniqueName. Otherwise, you may run into problems
-- with focus.
```

```
local setmetatable = setmetatable
local string = string
local awful  = require("awful")
local capi   = { mouse = mouse,
         screen = screen,
         client = client,
         timer = timer }

-- I use a namespace for my modules...
module("quake")

local QuakeConsole = {}

-- Display
function QuakeConsole:display()
   -- First, we locate the terminal
   local client = nil
   local i = 0
   for c in awful.client.cycle(function (c)
                  -- c.name may be changed!
                  return c.instance == self.name
                   end,
                   nil, self.screen) do
      i = i + 1
      if i == 1 then
     client = c
      else
     -- Additional matching clients, let's remove the sticky bit
     -- which may persist between awesome restarts. We don't close
     -- them as they may be valuable. They will just turn into a
     -- classic terminal.
     c.sticky = false
     c.ontop = false
     c.above = false
      end
   end

   if not client and not self.visible then
      -- The terminal is not here yet but we don't want it yet. Just do
nothing.
      return
   end

   if not client then
      -- The client does not exist, we spawn it
      awful.util.spawn(self.terminal .. " " .. string.format(self.argname,
self.name),
               false, self.screen)
      return
   end
```

```
   -- Comptute size
   local geom = capi.screen[self.screen].workarea
   local width, height = self.width, self.height
   if width  <= 1 then width = geom.width * width end
   if height <= 1 then height = geom.height * height end
   local x, y
   if     self.horiz == "left"  then x = geom.x
   elseif self.horiz == "right" then x = geom.width + geom.x - width
   else   x = geom.x + (geom.width - width)/2 end
   if     self.vert == "top"    then y = geom.y
   elseif self.vert == "bottom" then y = geom.height + geom.y - height
   else   y = geom.y + (geom.height - height)/2 end

   -- Resize
   awful.client.floating.set(client, true)
   client.border_width = 0
   client.size_hints_honor = false
   client:geometry({ x = x, y = y, width = width, height = height })

   -- Sticky and on top
   client.ontop = true
   client.above = true
   client.skip_taskbar = true
   client.sticky = true

   -- This is not a normal window, don't apply any specific keyboard stuff
   client:buttons({})
   client:keys({})

   -- Toggle display
   if self.visible then
      client.hidden = false
      client:raise()
      capi.client.focus = client
   else
      client.hidden = true
   end
end

-- Create a console
function QuakeConsole:new(config)
   -- The "console" object is just its configuration.

   -- The application to be invoked is:
   --   config.terminal .. " " .. string.format(config.argname, config.name)
   config.terminal = config.terminal or "xterm" -- application to spawn
   config.name     = config.name     or "QuakeConsoleNeedsUniqueName" --
window name
   config.argname  = config.argname  or "-name %s"     -- how to specify
window name
```

```
   -- If width or height <= 1 this is a proportion of the workspace
   config.height   = config.height   or 0.25           -- height
   config.width    = config.width    or 1          -- width
   config.vert     = config.vert     or "top"        -- top, bottom or
center
   config.horiz    = config.horiz    or "center"       -- left, right or
center

   config.screen   = config.screen or capi.mouse.screen
   config.visible  = config.visible or false -- Initially, not visible

   local console = setmetatable(config, { __index = QuakeConsole })
   capi.client.add_signal("manage",
               function(c)
                  if c.instance == console.name and c.screen ==
console.screen then
                  console:display()
                  end
               end)
   capi.client.add_signal("unmanage",
               function(c)
                  if c.instance == console.name and c.screen ==
console.screen then
                  console.visible = false
                  end
               end)

   -- "Reattach" currently running QuakeConsole. This is in case awesome is
restarted.
   local reattach = capi.timer { timeout = 0 }
   reattach:add_signal("timeout",
                function()
               reattach:stop()
               console:display()
                end)
   reattach:start()
   return console
end

-- Toggle the console
function QuakeConsole:toggle()
   self.visible = not self.visible
   self:display()
end

setmetatable(_M, { __call = function(_, ...) return QuakeConsole:new(...)
end })
```

This only works for applications that accept to be given a name through the command line (xterm, rxvt). Example of use :

```
local quake = require("quake")

local quakeconsole = {}
for s = 1, screen.count() do
   quakeconsole[s] = quake({ terminal = config.terminal,
                  height = 0.3,
                  screen = s })
end

config.keys.global = awful.util.table.join(
   config.keys.global,
   awful.key({ modkey }, "`",
         function () quakeconsole[mouse.screen]:toggle() end)
```