1. Unsuccessful attempt at (https://github.com/MonoGame/MonoGame/issues/5417) the full-screen command of a game engine project causes a null reference exception.
1. Step: trying to reproduce the bug.
   1. Looking trough the documentation of the software for instructions on how to install it.
   2. After not finding them in the read me or the documentation I tried to compile the projects in the source code, but they were not run-able as the project is a extension for Visual studio. Installing the current stable version of the project worked, but the bug was not reproducible.
   3. After looking through some of the sub folders I found an instruction on how to run the current version of the project, but the bug was not reproducible.
   4. Working under the assumption that no one had mentioned the fixing of the bug(which seemed not unlikely, as the issue tracker doesn't even have a bug-keyword) I cloned the repository and set it to the last commit before the bug was reported. The bug proved to be not reproducible. It might be possible that the bug was on the windows 10 platform for mobile applications and the reporter didn't mention this, or the bug might depend on some factor not mentioned in the bugs description.

2.As agreed I selected a different bug, as the first one proved to be beyond my ability to reproduce. The chosen bug was (url) a bug in a real time strategy game allowed to build walls outside your area of influence.

Description: By the game rules if you place a wall within 5 squares of another of the same type and the path is not blocked both will be connected. The bug occurs if the are between the first and the second wall is outside of the players area of influence(the are in which he can build) the game correctly displays the are as not build-able, but builds it if the second wall is placed.

Analysis: First I looked into the code till I found the class responsible for the placing of buildings. After that I looked for the code concerning line buildings(which include walls, barbed wire and similar). Then I observed the behavior of the code in debug mode while I placed walls, Both in the normal use case and the bug case.

Fixing: I compared the line-building with the normal building and Isolated the method used to check if a building could be placed on a given spot(actually two methods, one to check if the area is build-able and one to check if its in the current player's area of influence. I added a line of code to check if the wall section to be placed was within the area of influence(the code highlighted in green).

PlaceBuilding.cs line 79

```
foreach (var t in BuildingUtils.GetLineBuildCells(w, order.TargetLocation,
order.TargetString, buildingInfo))
                    {
                        if (self.World.CanPlaceBuilding(order.TargetString, buildingInfo, t,
null) &&buildingInfo.IsCloseEnoughToBase(self.World, order.Player, order.TargetString, t))
                        {
                            var building = w.CreateActor(order.TargetString, new
TypeDictionary
                        {
                            new LocationInit(t),
                            new OwnerInit(order.Player),
                            new FactionInit(faction)
                        });
```

This fix was only a partial success, as it was still possible to place walls if part of the 5 square line is outside of the are of influence, creating a partially completed line of wall. To fix this I modified the class that checks for another unit of the same type within 5 squares in a straight line, adding a check to see if all of the squares are within a players(not the local players, as that making causing problems with locally hosted AI opponents) treating the outside of area of influence like an obstacle.

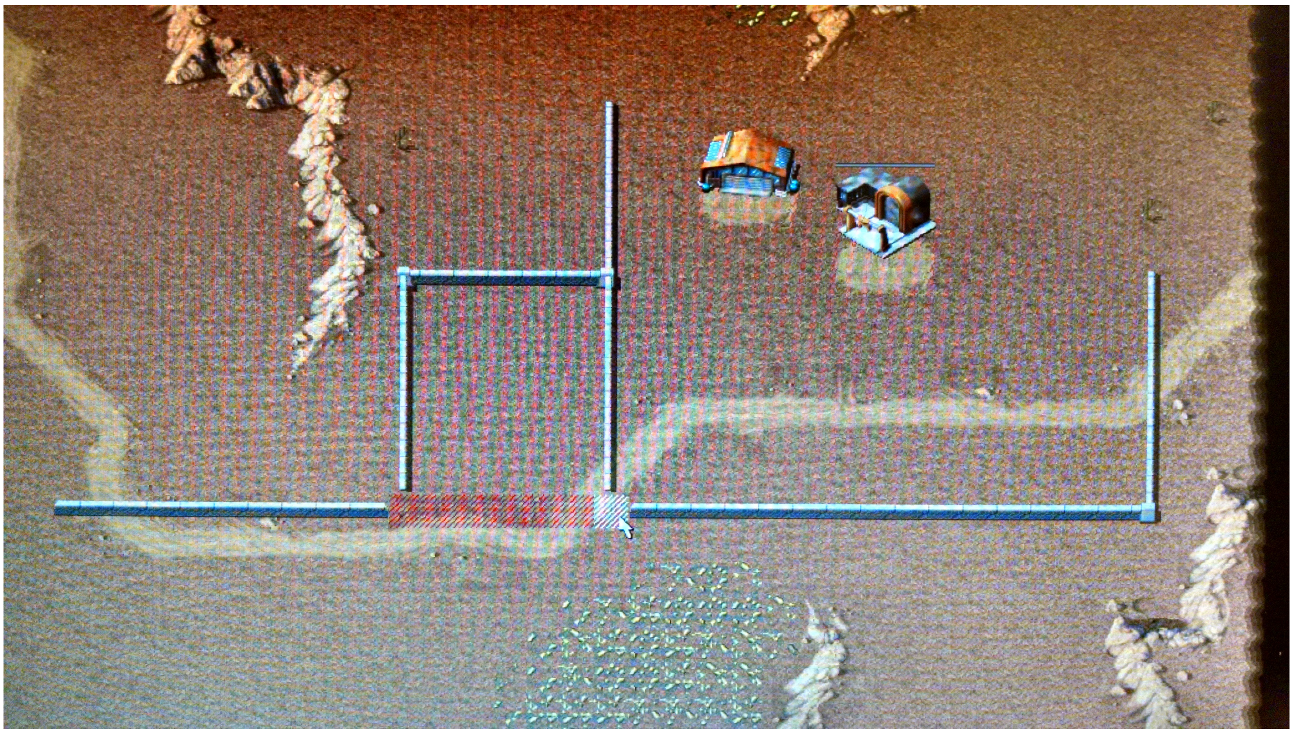
BuildingUtils.cs line 70

```
                              var cell = topLeft + i * vecs[d];
               bool influence = false;
               for(int j=0;j< world.Players.Length; j++)
               {
                   if(bi.IsCloseEnoughToBase(world, world.Players[j],name,cell))
                   {
                       influence = true;
                   }
               }
                              if (world.IsCellBuildable(cell, bi)&&influence)
                                     continue; // Cell is empty; continue search
```
Highlighted in green is the new code, that introduces a Boolean variable that is true if the observed cell is within a players area of influence.

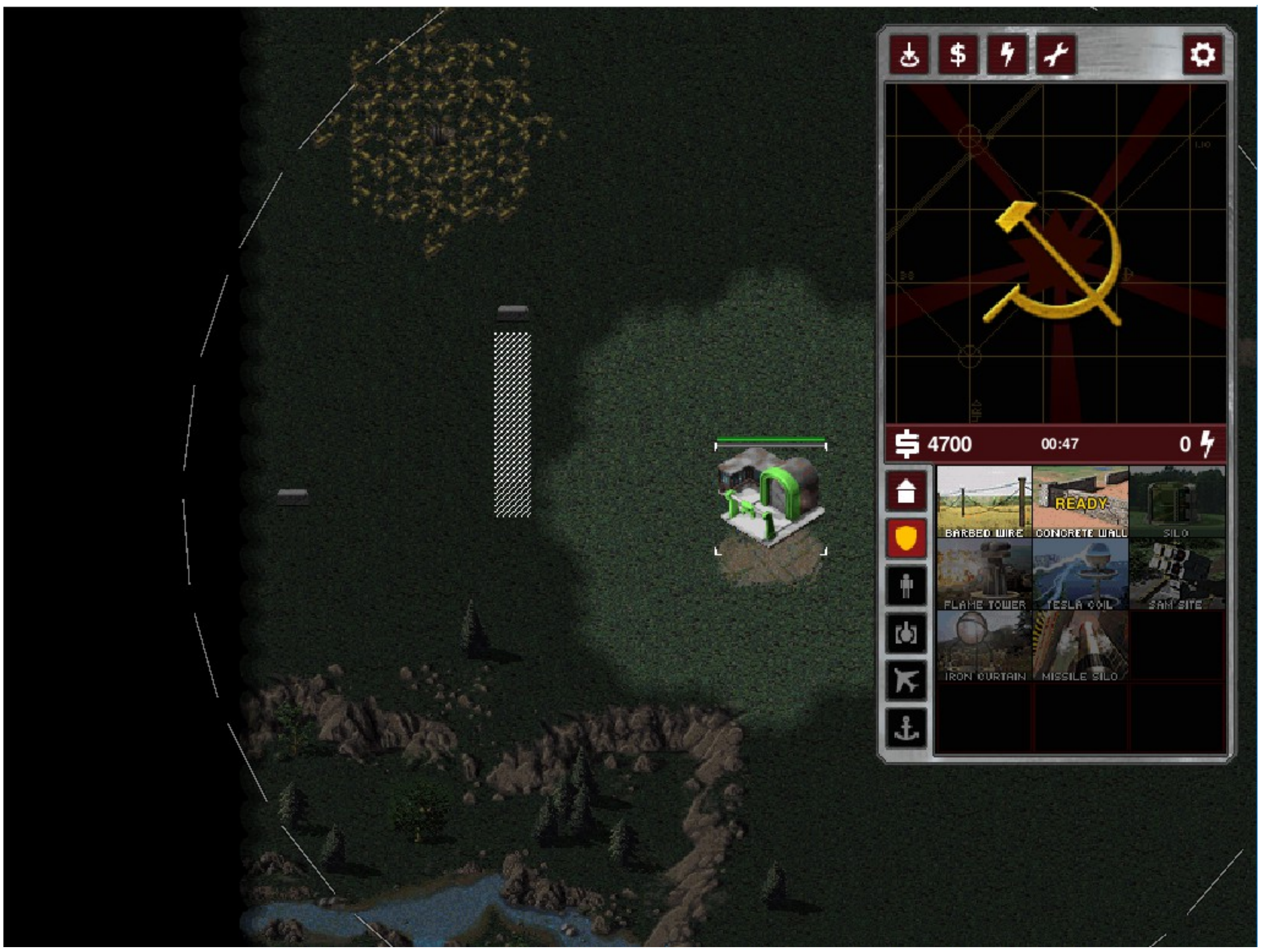
Both modifications where tested multiple times for different configuration of walls, and for all of them no bug occurred.

Bug Behaviour:

Modified:



Modified:


4700
00:47
0
BARBED WIRE
READY
CONCRETE WALL
SILO
FLAME TOWER
TESLA COIL
SAM SITE
IRON CURTAIN
MISSILE SILO