```haskell
-- vismol2-simple.hs

import Data.String
import Data.List
import Data.IORef
import Graphics.UI.GLUT
import System.Environment
import System.Exit ( exitWith, ExitCode(ExitSuccess) )

main = do
    -- there may be many command line arguments in various formats (e.g. -screen 0 in X),
    -- that's why we are looking for any .mol2 or .MOL2 suffix
    args <- getArgs
    let fileNames = filter (\ fn -> or [isSuffixOf ".mol2" fn, isSuffixOf ".MOL2" fn] ) args
    theText <- readFile $ fileNames !! 0 -- take 1st
    let theLines = lines theText
        theSplittedLines = map words theLines
        theAtoms = filter (\ aSplittedLine -> length aSplittedLine == 9) theSplittedLines
    atomsRef <- newIORef theAtoms
    -- glut --
    -- ($=) is GLUT- and IORef-specific assignment operator of type
    -- ($=) :: HasSetter s => s a -> a -> IO ()
    getArgsAndInitialize
    initialDisplayMode $= [ SingleBuffered, RGBMode, WithDepthBuffer ]
    initialWindowSize $= Size 300 300
    initialWindowPosition $= Position (-1) (-1)
    createWindow  $ fileNames !! 0
    displayCallback $= (display atomsRef)
    windowWidthRef  <- newIORef (0::GLint)
    windowHeightRef <- newIORef (0::GLint)
    reshapeCallback $= Just (reshape windowWidthRef windowHeightRef)
    xRef <- newIORef (0::GLint)
    yRef <- newIORef (0::GLint)
    keyboardMouseCallback $= Just (keyboardMouse xRef yRef)
    motionCallback $= Just ( motion atomsRef xRef yRef windowWidthRef windowHeightRef)
    clearColor $= Color4 0 0 0 1
    shadeModel $= Smooth
    materialAmbient Front $= Color4 1 1 1 1
    lighting $= Enabled
    position (Light 0) $= Vertex4 (-1) 1 1 0
    light (Light 0) $= Enabled
    depthFunc $= Just Less
    {- closeCallback $= Just (exitWith ExitSuccess) -- present in freeglut only -}
    mainLoop

keyboardMouse xRef yRef key@(MouseButton LeftButton) state@(Down) _ position@(Position x y) = do
    xRef $= x
    yRef $= y
keyboardMouse _ _ (Char '\27') Down _ _ = exitWith ExitSuccess -- on ESC, see closeCallback
keyboardMouse _ _ _ _ _ _ = return () -- to avoid crash?!

motion atomsRef xRef yRef windowWidthRef windowHeightRef position@(Position x y) = do
    theAtoms <- get atomsRef
    x0 <- get xRef
    y0 <- get yRef
    width  <- get windowWidthRef
    height <- get windowHeightRef
    let dx = fromIntegral (x - x0)
        dy = fromIntegral (y - y0)
        w = fromIntegral width
        h = fromIntegral height
        angle1 = (180::Float) * dx / w
        angle2 = (180::Float) * dy / h -- rather simplistic, but it works, really!
    clear [ ColorBuffer, DepthBuffer ]
    rotate angle1 $Vector3 0 (1::GLfloat) 0 -- ok, direction is correct
    rotate angle2 $Vector3 (1::GLfloat) 0 0 -- ok, direction is correct
    renderAtoms theAtoms
    flush
    xRef $= x
    yRef $= y

display atomsRef = do
    theAtoms <- get atomsRef
    clear [ ColorBuffer, DepthBuffer ]
    renderAtoms theAtoms
    flush

renderAtoms [] = do return ()
renderAtoms (atom:atoms) = do
    renderAtom atom
    renderAtoms atoms

renderAtom (_:_:x:y:z:atomType:_:_:charge) = do
    let dx = (read x)::GLfloat
        dy = (read y)::GLfloat
        dz = (read z)::GLfloat
    materialDiffuse Front $= atomColor4 atomType --Color4 1 1 1 1 --atomColor4 atomType
    translate $ Vector3 dx dy dz
    renderObject Solid (Sphere' (vdwRadius atomType) 32 32)
    translate $ Vector3 (-dx) (-dy) (-dz)

reshape windowWidthRef windowHeightRef size@(Size width height) = do
    windowWidth  <- get windowWidthRef  -- old value not used
    windowHeight <- get windowHeightRef -- old value not used
    viewport $= (Position 0 0, size)
    matrixMode $= Projection
    loadIdentity
    let wf = fromIntegral width
        hf = fromIntegral height
    if width <= height
        then ortho (-10) 10 (-10 * hf/wf) (10 * hf/wf) (-10) 10
        else ortho (-10 * wf/hf) (10 * wf/hf) (-10) 10 (-10) 10
    matrixMode $= Modelview 0
    -- loadIdentity
    windowWidthRef  $= width
    windowHeightRef $= height

vdwRadius atomType
    | atomType == "H"  = 1.20
    | atomType == "F"  = 1.47
    | atomType == "Cl" = 1.75
    | atomType == "Br" = 1.85
    | atomType == "I"  = 1.98
    | isPrefixOf "C." atomType = 1.70
    | isPrefixOf "N." atomType = 1.55
    | isPrefixOf "O." atomType = 1.52
    | isPrefixOf "S." atomType = 1.80
    | isPrefixOf "P." atomType = 1.80
    | otherwise = 2.0

atomColor4 atomType
    | atomType == "H"  = Color4 1 1 1 1 -- white
    | atomType == "F"  = Color4 0 1 0 1 -- green
    | atomType == "Cl" = Color4 0 1 0 1 -- green
    | atomType == "Br" = Color4 0 1 0 1 -- green
    | atomType == "I"  = Color4 0 1 0 1 -- green
    | isPrefixOf "C." atomType = Color4 0.5 0.5 0.5 1 -- gray
    | isPrefixOf "N." atomType = Color4 0 0 1 1 -- blue
    | isPrefixOf "O." atomType = Color4 1 0 0 1 -- red
    | isPrefixOf "S." atomType = Color4 1 1 0 1 -- yellow
    | isPrefixOf "P." atomType = Color4 1 0 1 1 -- magenta
    | otherwise = Color4 1 0 1 1 -- magenta
```